促进软件开发领域知识与创新的传播

架构师

ARCHITECT

推荐文章 | Article

链家网吕毅专访

蘑菇街七公专访

观点 | Opinion

Android开发者应该使用FlatBuffers替代JSON

我为什么选择Angular 2

专题 | Topic

IBM、Google、Oracle三巨头的公有云之殇

Geekbang 极客邦科技 | InfoQ

2016年8月

卷首语：重构的时机

重构并非难在怎么做，而是难在何时开始做。

对于一个高速发展的公司来说，停下来做重构从来就不是一个可接受的选项，“边开飞机边换引擎”才是这种公司想要的。当代码还不是很混乱的时候，重构的必要性不高，相比不小心重构出错让引擎熄火的风险来说，得过且过可能反而是一个明智之选。于是各种技术债就这样慢慢积累起来，直到业务因为各种技术债快跑不动的时候，架构师们才不得不用一些激进的重构手段快速的解决历史顽疾。如果重构获得了成功，大多数架构师在回顾过程的时候都会感慨：“要是早点重构就不会这么麻烦了”，不过在下一次重构到来之前，永远没人知道“早点”究竟是何时，同样的感慨会反复被提起。

就没有什么办法找到最合适的重构时机么？可能真没有。不过通过评估重构收益可以早一点察觉到重构的必要性，从而至少能做到稍微“早点”。

重构的根本目的就是让业务能跑的更快，达到更高的开发效率。对于

一般工程项目而言，基本不会出现无法实现的需求，只要有充足的时间，需求总能被实现。在这里提到的“需求”包含业务中所有明确或潜在需要的东西，并不局限于产品需求，各种性能、可用性、柔性指标也都是需求的一种。很显然，在需求一定的情况下，开发效率越高所需要花费的时间或人力就会越少，业务就能跑的更快。

并不是任何的重构都能达到预期的效果，因为重构本身需要付出成本，重构之后的架构也不一定真的能提升开发效率，只有通过估算发现“重构净收益”为正才是重构的好时机。

“重构净收益”由以下公式定义：

重构净收益 = 旧架构开发效率 – 新架构开发效率 – 重构成本 – 迁移成本

公式中每个项目的解释：

- 旧架构开发效率：采用旧架构完成需求所需要的时间；
- 新架构开发效率：采用新架构完成需求所需要的时间；
- 重构成本：重构所需要的时间，这一般是一次性投入；
- 迁移成本：为了解决新架构带来的额外问题所花费的时间，包括新架构的Bug修复、业务测试回归成本、学习成本等。

重构净收益是一个长期收益，随着需求不断演进新旧架构开发效率带来的差异会日渐明显，直到能够超过重构成本和迁移成本，如果估算发现收益为正，那么意味着当前就是重构的好时机，应该着手准备了。

举个使用公式的例子。

假设有一个服务由于长期开发不注重代码复用，重复代码颇多，开发一个新功能，比如增加一种新的通用参数，就不得不修改几乎所有接口的代码，那么是否值得消除重复代码？按直觉来说，答案应该就是肯定的，但实际上并非如此，这些情况基本可以用“重构净收益”公式来解释。

如果这个服务基本不再维护，旧架构并不会持续的带来开发成本，假如重构成本大于使用旧架构完成需求的时间，那么重构收益就一定为负，重构不值得做；

如果重构会带来很大的迁移成本，比如会造成几十万行无人维护缺乏测试覆盖的旧代码发生改动，无人能够保证改动正确性，那么就算重构本身成本不高（假设能够利用一些脚本大批量抽取重复的代码），这种重构也是值得商榷的；

如果重构之后的新架构过于超前，学习门槛很高，当前团队很难可靠的掌握高效的开发方法，最终这些学习成本会被放在迁移成本之中，假如过大也意味着收益为负，重构不值得做或者必须换一种更让团队容易接受的方案。

限于篇幅，这里就不展开说明公式的更多用法，大家不妨拿自己工作中的例子套用试试，看是否有帮助。需要特别注意，就算重构净收益为正，也并不意味着应该停下业务做重构，重构带来的是长期收益而业务需求代表着短期收益，架构师有责任在确保业务正常运转的情况下为未来做准备。

滴滴平台产品中心技术总监

杜欢

CONTENTS / 目录

推荐文章 | Article

专访吕毅：链家网技术架构的演进之路

专访蘑菇街七公：25 倍增长远非极限，优化需要偏执狂

观点 | Opinion

为什么 Android 开发者应该使用 FlatBuffers 替代 JSON

我为什么选择 Angular 2

专题 | Topic

IBM、Google、Oracle 三巨头的公有云之殇（上）

IBM、Google、Oracle 三巨头的公有云之殇（下）

架构师

2016 年 8 月刊

本期主编 韩　婷

流程编辑 丁晓昀

发行人　霍泰稳

提供反馈 feedback@cn.infoq.com

商务合作 sales@cn.infoq.com

内容合作 editors@cn.infoq.com

CNUTCon 2016
全球容器技术大会

2016.9.9－10　北京 喜来登长城饭店

案 / 例 / 剖 / 析 · 实 / 践 / 驱 / 动

8折优惠（截止8月7日）团购报名更多优惠

主办方 Geekbang. 极客邦科技　InfoQ

最佳的容器实践案例都在这里

10大精彩议题

- 容器云专场
- 持续集成/持续交付
- 微服务
- Kubernetes实践
- Mesos实践
- Docker实践
- 金融场
- 创业场
- 电商场
- 综合场

超40位大牛讲师
解读最新、最前沿的容器技术

方国伟
平安科技云平台
事业部总经理
副总工程师

黄 成
上交所
资深架构师

梁启鸿
广发证券IT研发
董事总经理
首席架构师

王烈波
广发证券
资深架构师

联系我们 / CONTACT US

商务咨询：yolanda@infoq.com　13426412029

会务咨询：emma@infoq.com　15901094010

票务咨询：molly@c4media.com　15801572605

扫码进入聊聊架构社群

扫码进入容器大会官网

专访吕毅：链家网技术架构的演进之路

作者 木环

链家网虽然成立于 2010 年，但是其技术团队却于 2014 年正式创立。此前技术开发采用的是传统模式，每个业务都会单独地重新开发，不仅造成各个模块孤立，并且开发人力投入成本巨大。鉴于互联网时代企业业务发展迅速，原有的传统化方式已经不适用，链家网正式建立技术团队，在原有的传统架构基础上开始了优化工作。

团队对已有的业务进行抽象，将各个业务模块中的公共部分综合出来，据此添加了一层公共的服务层，实现了平台服务化，扩展技术基础能力。此外，链家还重新搭建系统监控并完成日志监控，双级监控完善技术运营能力。目前的技术架构充分满足业务发展情况。InfoQ 就技术架构变迁对链家网平台服务架构师吕毅进行了采访。

InfoQ：谈谈您为什么会选择加入链家？

吕毅：关于自我成长，相对于前两家我任职过的成熟互联网公司，在我入职时，链家网相对初期。这就预示着链家网这里会有很多挑战，我一

直清晰认识到自己是有惰性的，所以我希望找到一家技术挑战多的公司，这样这些挑战会刺激我兴奋、鞭策我成长。在这里的一年，的确验证的当初我的这个想法，刚入职是我与两位伙伴做话务平台项目，不满一年，我们平台架构组已是 8 人的团队，负责着公司 6 项平台化服务，挑战与成长并行，大家都感受到架构设计、技术选型与实现、上下游沟通上突飞猛进的成长。

关于职业选择，我是技术角色，顺应之前的职业发展，我在抉择机会时候还是会选择互联网公司。然而，我将互联网公司分为两类，纯互联网公司与传统企业 + 互联网公司。在纯互联网公司这座大山里，我作为技术角色站在过最好的山头，虽未登顶但也看清了这座山头的风景，转头看，最近些年才被关注的传统企业 + 互联网公司的这片山林，一片荆棘，而我具备的能力可以开垦这片大山，为了追求开垦的乐趣从而来了这里。至于为什么是链家，当时做选择时链家人为我描绘的链家蓝图吸引了我，当时了解到产品整体负责人闫觅之前是百度的高工，这让我也对链家好感增加不少，我想技术出身的 PM 在产品规划上应该会更理性些。也恰巧那会做选择时想咨询下前辈鸟哥意见，鸟哥说他下个月也去链家，这让链家的 Offer 加分不少。

入职前的种种对传统行业的顾虑，在入职后逐渐的全部解除了。这里提供的空间与自由度，管理的扁平程度，都是让我在入职前难以想象的。在链家网，时常在露台碰见闫觅、鸟哥，吐槽吐槽对产品的不快，对公司的发展的不理解，他们都会耐心的解释为何如此，从什么角度考虑做的决策，每次聊完后我都能自我纠正了想法，同时我也乐意将这些解答传递给我们的组员。链家网给予员工充分的空间，在保证自身工作的基础上，我可以与 HRBP 组织录制平台化服务系列课程、与员工发展部规划初级工程

师培训、与外部技术社团联络到链家来组织技术活动、与一些公司技术团队做技术交流都得到了大家的肯定与支持。这里有足够的空间，只怕你的想象力不够。

InfoQ：谈谈目前链家网的主要业务及团队规模？

吕毅：链家网主要业务在很多公开资料中都有谈到，二手房方向是链家网目前的重心，围绕二手房开展的面向业主与客户的链家网、链家APP，面向链家十万多经纪人的 Link 项目，都是链家网目前的重点业务。链家网目前员工 1000 人，产品技术角色占比达一半。

InfoQ：您提到链家网这两年在架构上做了很多的调整，可以谈谈链家网的架构演进历程吗？

吕毅：链家网的架构演化的确从未停歇，从技术团队建立至今的两年时间，架构上大致分为两个阶段，如图 1。

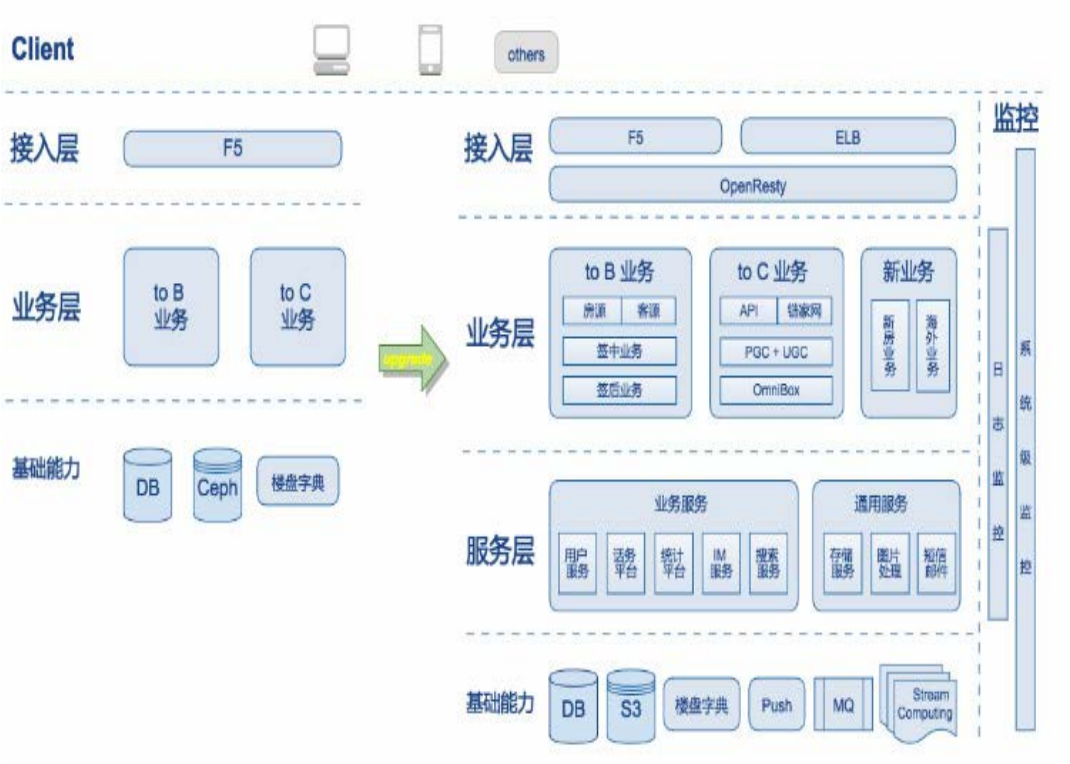


图 1

2014~2015 年技术上关注两件事，一块是将链家集团曾交付外包实现的面向经纪人业务改造为自主研发，另一块是从零打造面向用户的业务链家网与掌上链家 APP（目前已改名为链家 APP），即第一年的技术架构重在业务的建设。

2015 年至今，业务的逐渐成熟，引入了新的挑战。业务方向内的子业务细化伴随着链家新业务的开展，此时迫切希望抽离公共技术部分，避免重复造车的同时也希望由公共服务来支撑好业务线发展，让业务线更好的满足产品迭代。从 2015 年开始，一系列的公共服务的建设，构建成了整体技术架构上的服务层，同时也扩展了公司基础技术能力，并推动着建设起了系统、日志级的监控。

InfoQ：您对目前的架构满意吗？下一步优化的方向是什么？

吕毅：从技术架构的演化可见链家网成立的两年间，技术架构从无到有、从有到完善，一切都在快速的发展着。我们目前的技术架构是充分满足当前的业务发展情况的。要说对架构是否有不满，的确是有的。如今链家网业务各个方向发展迅猛，技术架构上的压力不小，理想状态是希望技术架构在支撑业务时，时刻游刃有余，但这会是条任重道远的路。

下一步的调整与优化方向，还是配合业务上一起做好服务化。业务层面希望将目前打包在一起的功能逐渐服务化，而技术架构上需要提前调整与优化，提供服务维护、服务治理、服务监控、服务通信等一系列围绕产品服务、技术服务的周边技术支持，这块是明确的方向。然而，一些未知的方向，还得与业务线、管理者们常沟通，不断摸索，按需计划着开展。

InfoQ：服务层的优化已经进行一年多，这过程中有什么经验可以和大家分享的吗？

吕毅：在来链家网前到现在，算起来的确有一年多在做平台化、公共

服务这些事情，伴随着自我成长，感慨的确很多，列三点深刻感受的在这里与大家共勉。

第一，服务源自需求。只有业务需要的才是值得做的，只有多个业务线都需要的，才是值得拿出来做成平台服务的。多去关注业务，寻找业务线研发团队共性部分，才能发现需求，通过与业务线的沟通才能发现做平台服务的价值，所以做平台服务的技术角色切忌闭门造车，只有把自己当做半个业务线的 RD 才能感受到业务线的痛点。

第二，“第三选择”。常见行业内做公共服务的同学与业务线同学有些碰撞，双方各持观点，对一些边界问题拿捏不清。此时大家秉持第三选择，一起寻找更好的解决办法，目标达成一致，问题便可化解。当然第三选择的解决办法用在哪里都是合适的。

第三，服务的解耦。这是平台化服务的基本原则，服务间的解耦、服务内部的功能解耦，都是在日常设计、开发中需要注意的，只有将一块事情拆成多个点，才好做点与点之间的联系，以有限的功能点构建出无限的能力。这是老生常谈的点了，但依然值得再提一遍。

InfoQ：请讲讲你们的日志分析平台架构吧？目前的日志处理流程是怎么样的？处理的日志大概是怎样的数量级？

吕毅：日志分析平台在前面技术架构图中，属于纵向的监控部分中的“日志监控”环节，主要解决业务模块、服务模块日志字段的数据收集、展示、监控。架构设计引用公开资料中的截图来说明（见图 2）。

日志通过 Kafka 收集，根据日志所属 RD 配置的统计、监控规则通过 Apache Storm 实时分析日志，并将结果集数据存储数据库，实时分析期间若触发了监控规则阈值，则触发报警。数据库中的数据可以做实时的数据展示，整套方案可以让研发、测试角色实时查看日志情况，避免了大家

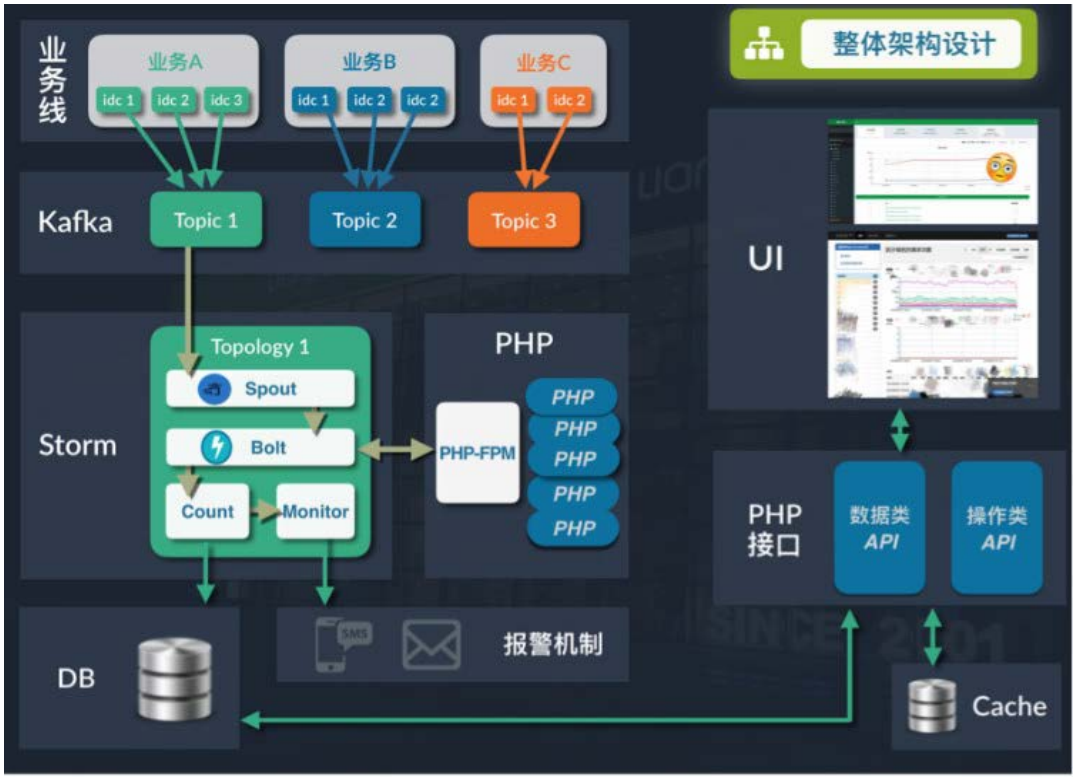


图 2

日常合并日志文件再做 shell 统计的问题，并提供平台可以持续使用。

目前日志平台每秒处理的 30 万行日志，处理结果的展示与报警延迟在 2s 以内，并且这套解决方案有计划在后续开源，让业内同学低学习成本掌握并构建到生产环境中。

InfoQ：要设计一个高可用、高吞吐的日志平台，您认为需要重点考虑和解决哪几个方面的问题？

吕毅：设计这样一套日志方案，有以下几点需要关注：数据的收集、数据的处理、数据的存储、数据的展示。

其中数据的收集部分行业内部不少用 flume 采集日志，我们最早期的 beta 版本也是如此，但在我们方案中缓存日志数据的组建 Kafka 在年初版本中发布了 Kafka Connector 功能，实现了类似 flume 能力，故我们上线前就改用 Kafka 全套解决日志收集问题。

数据的处理，因为要求实时性，行业内也有两套方案，Spark Streaming 与 Apache Storm。两者共性很多，但选择 Storm 的原因主要是 Storm 的设计与 Feature 更专注实时运算，而 Spark 做离线的大并发流式处理是不错的，例如流式批运算、图片处理等等。

数据的存储，选型比较多，大家做 DB 选型时需关注企业级日志系统需要有大批量数据写入，特别是业务高峰时期，那么选择一个良好支持高并发写入特性的 DB 即可，我们使用的 HBase。

数据的展示，这块就比较灵活，根据自己需求，从 DB 中选择数据通过组织拼装成格式化数据，配合上前端特效展示即可。

InfoQ：在你们的平台中，日志数据会采集哪些信息？你们是如何统一其他开发人员的日志格式和信息的？

吕毅：如日志平台的架构图所示，数据的采集基本是全量的，日志文件大部分是研发关注的信息，这些被关注的日志文件都会被收集，用作实时分析计数。

在日志格式这块，我们在接入层做了统一日志格式，故这份日志将会是所有业务的请求日志全集。对于业务模块、服务模块自己的日志，我们会给予建议，但没有强制规定日志格式，这部分有差异的日志格式，会在日志平台中研发角色配置统计规则时，通过正则匹配自助扣取想要的日志字段用作统计、监控。

InfoQ：与日志监控相比，系统级监控对业务层、服务层的监控指标及目的是怎样的？

吕毅：系统监控和日志监控比较，有这么几点差异。

- 面向人群：

系统监控，主要面向运维角色，RD 角色关注较少，RD 对于系统监控，

更关注报警；日志监控，主要面向 RD、QA 角色，他们关注业务日志某些字段出现的频次、某些值的最大最小值等，同时也可以设置基于日志的业务报警。

- 数据源：

系统监控，更多收集 syslog、机器设备数据；日志监控，主要手机业务线自己打的日志内容。

两者互补，在需要了解业务所属的服务器信息时候，在系统监控上查看；想了解业务数据情况时，在这套日志监控平台上看。

InfoQ：链家也是一家从传统公司转型为互联网公司的代表，你认为这中间，最大的挑战是什么？你有什么经验分享？

吕毅：传统公司转型互联网，我认同链家集团老总发表的观点，这样的转型并不是将原有业务打死，全部线上化，而是传统公司加入互联网属性，是传统业务与互联网业务相结合的过程。链家网闫觅也曾说过“这是一次人文革命”，互联网更多的是一个工具，一个新型的商机渠道，一个提升效率的系统方案。

那么链家在做线上线下结合的过程中，最大的挑战应该有两点：传统业务的梳理与互联网化改造，线上业务与传统业务的融合。

传统业务的梳理与互联网化改造，链家网从创立之初就在践行，至今仍是重点工作之一，可见难度并非一般。在几次与经纪人业务侧的 PM 交流后，深感房产交易的复杂性，就北京而言交易中的一个环节可能会有 30 多种角色参与，全国各个城市政策不一，更是难上加难，而链家网在努力优化流程，给业主、用户提供清晰明了的房产业务体验。这方面的挑战折射在技术上，就是复杂的业务流程控制、风控体系、数据安全等技术难点。

线上业务与传统业务的融合，存在于在 To C 业务与 To B 业务存在交集的部分。To C 业务是我们所长，但如何让还未进入房产交易流程内的普通用户，快速了解链家房产信息、交易流程，让普通用户在有需求时第一时间通过链家网、掌上链家 APP 发起沟通、交易，这都是挑战。折射在技术上就是如何保证好线上用户如何快速、满足需求的查找到信息、如何与经纪人快速建立有效沟通、如何沉淀意向客户并建立起关系等等的技术问题。

作为链家网平台架构组负责人，与组内 7 位伙伴一起，对技术追求极致的同时，我们也时刻关注着业务线的难点、痛点，希望通过技术手段提供支持与帮助，我想所有链家网角色的信念是一致的：让房产交易不再难，提供诚信便捷的服务体验！

专访蘑菇街七公：25 倍增长远非极限，优化需要偏执狂

作者 李东辉

每秒能支撑的峰值订单数是衡量电商系统高并发可扩展能力的重要体现，在电商内提升每秒支撑订单数存在无数的方法，每一个方法都存在各自的优化角度和对应的障碍挑战，如何在一开始根据自身企业的特点选择最合适的优化路径，并其后在这路径上高效贯彻和执行，对于团队尤其是初创团队而言都不是简单的事情。

现在我们就来采访七公老师，看看蘑菇街是如何快速走出一条高效、实用的服务化发展路径的。

InfoQ：在 2014 年以前您在阿里巴巴 B2B 负责 Aliexpress 资金中心等项目，能否谈谈在阿里的这段经历给您之后出国游历、回国加入蘑菇街带来了哪些动因和影响？

七公：我是 2011 年 3 月加入阿里巴巴的，彼时阿里整个的业务、技术体系都处在一个飞速发展的时期，我个人也获得了飞快的成长。受去看世界的梦想推动，2014 年 8 月我和女朋友一起从阿里辞职，出国游历了半年。

回来之后，阿里当时内部各方面整体已经趋向稳定，我希望找到另一家飞速发展的公司，北京上海找了一圈不满意，最终还是回到了杭州，加入了蘑菇街。在蘑菇街一年的时间里，我们业务和技术都获得了火箭般的飞跃式前进，我一年里的进步和收获比在其他公司呆两年得到的还要多，实现了和公司的共进步、同成长。

InfoQ：您加入蘑菇街电商团队后带领团队同学仅用一年便完成服务架构的各阶段升级，能否谈谈你们是如何规划优化路径并高效实现的？在高效率工作上你们有什么经验或技巧可以分享？

七公：第一个阶段【蘑菇街系统拆分 & 服务化 1.0 体系构建】，是我们做 PHP 全面转型 Java 体系、初步建成电商基础服务中心的战略规划，在面临不停歇的业务需求和巨大的系统改造压力下，我们采用瀑布流工程方法，通过规划、分析、设计实现、测试、服务产品 & 文档交付的过程，高质量地把第一阶段的服务化建设根基像打桩一样打牢，然后通过进一步的迭代开发不断完善。

第二阶段【购买链路核心服务的性能提升 & 服务架构 1.5】和第三阶段【服务 SLA 保障推动稳定性提升 & 服务架构 2.0】，实际上是业务迅猛发展、流量不断上涨、日常和大促稳定性保障的强烈需求推动我们自身服务架构的升级。我们通过引入 Scrum 的敏捷开发模式，项目中的每个人都是猪（敏捷开发中，猪为项目负责人，鸡只是普通参与者，寓意来自猪要牺牲才能提供食物而鸡只要下个蛋就行了），每个人都要为服务框架升级和项目进展负责。

我们先后有十人以上共同推进了服务框架负载均衡、降级限流、连接切换优化等基础框架演进，以及监控、服务端超时控制、多版本、分组隔离、动态路由等服务治理体系完善。

总结一下：高效推进上，我们的经验是首先要采用项目的方式进行管理，再一个是时机不同、阶段不同，选择的项目工程方法也不一样：

- 新架构探索：建议采用传统的瀑布式和迭代式开发；
- 架构的持续迭代：可以尝试一些新的开发方法。

InfoQ：能否介绍蘑菇街系统架构的中间件系统？蘑菇街的一系列中间件是如何实现 Web 应用层和基础设施层对接的，各中间件如何确保在全站稳定落地？

七公：图 1 为蘑菇街购买链路服务架构示意图。

从蘑菇街购买链路服务架构 1.0 到 1.5，我们完成了服务化框架 Tesla、分库分表 TSharding、分布式缓存 MoguCache、Corgi MQ、配置中心 Metabase、调用链跟踪 Lurker 等一系列重要中间件的自研、完善和在

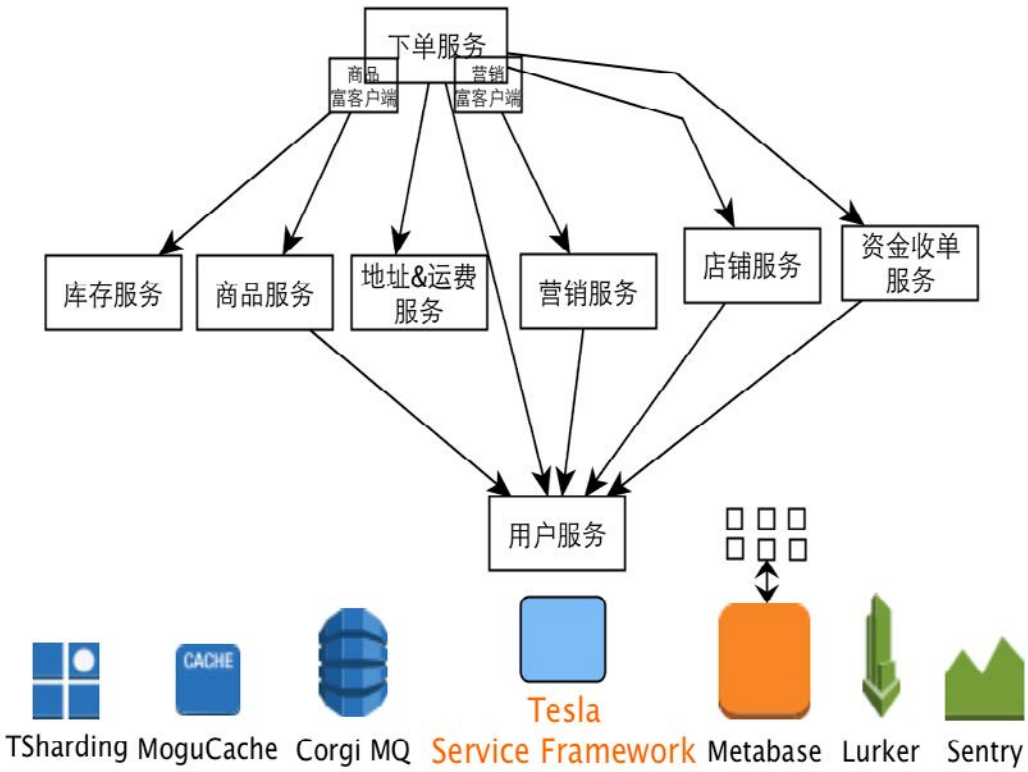


图 1

全站落地。这一系列中间件的研发，是随着业务系统和服务框架的不断发展（最早我们只有 Tesla 一个中间件）而逐步衍生出来的。

2015 年初，蘑菇街全网还在 PHP 体系，Tesla 框架诞生后，我们先构建了用户、商品、交易、促销等一系列的电商基础服务中心：

- Tesla框架支撑了PHP Web应用到服务中心的PRC调用场景；
- 服务中心内部有数据容量提升需求，所以先后有了TSharding和Raptor分库分表中间件；
- 服务内部有异步化处理需求，有了Corgi MQ；
- 服务中心内部有分布式缓存需求，所以有了MoguCache等。

所以服务框架是支撑和推动整个网站架构发展的核心和源动力。2015 年底我们启动了 PHP Web 单体应用拆分为一个个 Java Web 业务应用和前后端分离项目，很快购买链路完成了 PHP 体系到 Java 生态的彻底升级。

Tesla 走向真正的服务框架（而不仅仅是一个 RPC 框架）是随着业务的不断发展和业务系统的强烈要求开始的。最早业务系统面临服务发布时不平滑（客户端几秒甚至数 10 秒后才拉到更新的 IP 列表）等问题，以及业务服务蓬勃发展时期强烈的服务治理需求，Tesla 不断进行服务框架的改造升级：

- 通过新的配置中心Metabase的诞生和客户端拉取配置优化、客户端响应服务端连接关闭事件解决连接优化问题；
- 通过对监控预警、动态限流、服务端超时、服务分组等的支持完善了服务治理体系。

实际上业界的众多服务框架也是这么一步步发展起来的，没有捷径。

中间件在全站的有效落地，是靠中间件团队和业务系统支撑团队密切配合完成的。新的中间件刚刚开发出来，肯定会有各种问题出现，但是中

间件也是业务系统的实际需求推动才产生的，所以双方的目标、利益点是相同的。我们会采用项目的方式，每个项目成员都对中间件的使用推广和自身完善负责，共同推动中间件在全站的稳定落地。

InfoQ：在中间件研发过程中，开源技术起到怎样的作用？

七公：蘑菇街是拥抱开源技术的。在我们的技术体系中，很大一部分基础技术组件依赖于开源产品。我们的 Tesla 服务框架是基于 Netty 开发的，最早我们用的 MQ 是 RabbitMQ，后来我们才转向自研；我们的 Sentry 监控平台是基于 Grafana；目前我们仍然在广泛使用 Kafka 来收集全站日志，使用 Zookeeper 协调分布式系统，使用 Hadoop 生态来支撑数据平台计算任务，等等。

而自研 Tesla、Corgi MQ 等等中间件的原因是因为这些是支撑我们网站发展的核心产品，我们必须要能够完全掌控、有能力做到深度定制，以便快速支持业务发展，而不是成为业务发展的瓶颈和阻碍；而这些中间件的诞生、持续发展和在全站落地之后，变成了我们技术团队乃至整个蘑菇街的核心竞争力之一。

针对纯技术的中间件 / 产品，我们会选择性的开源。TSharding 是我们最早的分库分表中间件，之前因为有小米、有赞等不少技术同仁都有咨询，所以我们开放在了 Github 上，见 https://github.com/baihui212/tsharding。

InfoQ：如何通过数据分析来指导“前瞻性地谋划实施、支撑业务快速发展”？在电商中哪些数据发生怎样的变化会给系统架构带来预警，提示需要改良或优化？能否结合谈谈你们的峰值系统的监控架构和方案？

七公：“前瞻性地谋划实施”可以通过关注业务数据和系统数据的增长变化情况来获得一个合理的度。通过当前业务的增速能推算出来三五年

内的业务爆发和数据增长情况，如果有些架构上的不适宜（如过度中心化或存在易攻击点、系统难以扩展或者稳定性难以保障等问题），要尽早规划新的架构形态并快速推进实施，提早解除可能阻碍业务发展的绊脚石。但是也不能过度设计，考虑五年以后、十年以后的情况做设计，可能还未到时候，现有架构早就过时了。

电商数据比较复杂，概括如下：

- 业务数据

目前我会关注 DAU，分平台的 GMV、UV、客单价、支付订单数、支付用户数等。

- 系统数据

主要是容量和系统运行情况，容量包括数据库表空间、磁盘使用情况、服务应用的水位、缓存集群的分片情况等；系统运行状态主要包括 QPS、RT、调用成功率、CPU 占用率、内存使用、IO 等。

只要持续关注这些核心数据，才能敏锐地把控到系统接下来的发展趋势和改造方向。

蘑菇街大促峰值的监控分两部分：一部分是实时业务监控，播报实时 PV、UV、购买订单数、GMV、客单价的情况，这部分是通过实时计算平台的流式计算任务来完成的，延迟在秒级左右；另一部分是实时系统监控，我们是通过异步上报、LogAgent 收集、实时统计分析来保证 10s 左右的延迟就能把系统的实时统计指标呈现在可视化监控平台上。前者给业务方密切关注，后者则是我们做大促保障指挥决策、预案执行的哨兵和自动化告警的主要手段。

InfoQ：目前蘑菇街的架构是否已经达到了“该优化的已经优化了”的阶段？您将在 ArchSummit 的分享中提到“25 倍”的优化路径，在您

的猜想下，25 倍是否是优化的极限，您认为短期未来还可以达到多少，如何做到？

七公：蘑菇街业务每年都保持 3 倍多的迅猛增长，对技术的挑战一直非常大。2015 年我们主要完成了业务应用服务化建设 & 服务架构升级、基础中间件研发落地、前后端分离、稳定性平台建设 & 大促保障常态化等重大技术改造，为网站架构的进一步发展打下了很好的基础。

目前我们还面临诸多问题和挑战，比如前端组件化的建设和新业务的快速交付、蘑菇街 / 美丽说 / 淘世界整个集团各平台业务发展的快速支撑和系统化的稳定性保障 & 质量保障能力；JDK8、异步框架等技术在全站的落地；应用高性能的持续优化、虚拟化 & 私有云的持续建设等等，蘑菇街还远未到“该优化的已经优化了”的阶段。我们要做的就是把每行代码都写好、每件事情都做到位，让我们的系统变的越来越强大，同时用更少的人力和机器成本去支持更大体量的业务。

关于优化的极限问题，最终追求的极限不是倍数，是在机器数上。机器堆的多了，自然可以支撑更大的流量，但是我们目前在做的是用更少的机器，支持更多倍的流量。目前我们主要还是进一步提高虚级化占比的方式（已逼近上限）和单机能力提升（还大有可为）的方式来进行的。

InfoQ：为什么您认为“根据场景来选用性能优化方案，没有通用方案”，蘑菇街以前是否考虑过“通用方案”的设计？能否结合蘑菇街架构升级案例谈谈这个体会？

七公：性能优化有很多种手段。

- 硬件

对硬件更新换代、提升硬件能力、调优硬件参数可以提升性能，横向扩展方面加机器也可以提升性能。

- 软件

对数据库 /Web Server 等软件进行配置优化、对 CPU 执行效率做优化、对 IO 效率做优化、对程序 CPU 计算复杂度做优化等等都可以提升性能，甚至仅仅做业务化简也能得到很可观的 RT 的降低。

2015 年初蘑菇街面临非常大的性能问题，只能支撑 400 单左右每秒的交易创建，严重不满足业务超高速发展和大促迅猛流量的要求。我们在完成系统拆分 & 服务化 1.0 体系构建后，开始了购买链路的专项性能提升项目。这个项目过程中翻越了许多障碍、篱笆，比如 DB 单点写瓶颈是一直压在头上的一座大山。

我们通过自研分库分表组件 TSharding，完成分库分表，最终下单服务支持了写入的无限水平扩展，订单库容量提升到了千亿级规模；营销服务的 RT 不稳定，每轮压测，营销服务都能再进一步，然后遇到下一个问题；我们通过 SQL 优化、缓存 & 预处理、读写分离等手段，大促期间最终计价接口 RT 稳定在 7ms；异步处理非常有必要性，能大大降低服务响应 RT，然后我们通过自己的方式解决异步化后的分布式事务问题，等等。

我们优化的案例还有很多，优化的过程很类似：找到瓶颈点——优化——压测验证然后再循环。但是最优的方案往往都是针对该场景和问题定制的，不应该去追求通用方案，也不会有通用方案。

InfoQ：并没有很多技术人员能有处理海量服务的机会，在从事这方面的工作中您有什么特别的感悟或经验可以和大家分享吗？

七公：我的体会是：只有自己经历过、尝试过的，才能真正成为自己的。如果有意愿深入接触高并发高可用高可扩展服务架构，但是目前还没有机会的，建议尽早换到业务在快速发展的有大流量场景的互联网公司，给自己挑战自我的机会。

为什么 Android 开发者应该使用 FlatBuffers 替代 JSON

作者 Amit Shekhar 译者 程大治

你可能会问，既然我们已经有很标准的 JSON 以及转换库比如 GSON 和 Jackson，为什么还要使用新的工具呢？

不妨先试一下 FlatBuffers，然后你就会发现它比 JSON 快得多。

FlatBuffers 是什么

FlatBuffers 是一个高效的跨平台序列化类库，可以在 C++、C#、C、Go、Java、JavaScript、PHP 和 Python 中使用。是 Google 开发的，是为了应用在游戏开发，以及其他注重性能的应用上。

为什么要使用 FlatBuffers

- 不需要解析/拆包就可以访问序列化数据 ：FlatBuffers与其他库不同之处就在于它使用二进制缓冲文件来表示层次数据，这样它们就可以被直接访问而不需解析与拆包，同时还支持数据

结构进化（前进、后退兼容性）。

- 内存高效速度快 — 访问数据时只需要访问内存中的缓冲区。它不需要多余的内存分配（至少在C++是这样，其他语言中可能会有变动）。FlatBuffers还适合配合mmap或数据流使用，只需要缓冲区的一部分存储在内存中。访问时速度接近原结构访问，只有一点延迟（一种虚函数表vtable），是为了允许格式升级以及可选字段。FlatBuffers适合那些花费了大量时间和空间（内存分配）来访问和构建序列化数据的项目，比如游戏以及其他对表现敏感的应用。可以参考这里的基准。
- 灵活：由于有可选字段，你不但有很强的升级和回退兼容性（对于历史悠久的游戏尤其重要，不用为了每个版本升级所有数据），在选择要存储哪些数据以及设计数据结构时也很自由。
- 轻量的code footprint：FlatBuffers只需要很少量的生成代码，以及一个表示最小依赖的很小的头文件，很容易集成。细节上可以看上面的基准页。
- 强类型：编译时报错，而不需要自己写重复的容易出错的运行时检查。它可以自动生成有用的代码。
- 使用方便：生成的C++代码允许精简访问与构建代码。还有可选的用于实现图表解析、类似JSON的运行时字符串展示等功能的方法。（后者比JSON解析库更快，内存效率更高）
- 代码跨平台且没有依赖：C++代码可以运行在任何近代的gcc/clang和VS2010上。同时还有用于测试和范例的构建文件（Android中.mk文件，其他平台是cmake文件）。

谁在使用 FlatBuffers

- BobbleApp，印度第一贴图App。我们在BobbleApp中使用FlatBuffers后App的性能明显增强。
- Cocos2d-x，第一开源移动游戏引擎，使用FlatBuffers来序列化所有的游戏数据。
- Facebook使用FlatBuffers在Android App中进行客户端服务端的沟通。他们写了一篇文章来描述FlatBuffers是如何加速加载内容的。
- Google的Fun Propulsion Labs在他们所有的库和游戏中大量使用FlatBuffers。

App 性能有多大提高

- 解析速度 解析一个20KB的JSON流（这差不多是BobbleApp的返回大小）需要35ms，超过了UI刷新间隔也就是16.6ms。如果解析JSON的话，我们就在滑动时就会因为要从磁盘加载缓存而导致掉帧（视觉上的卡顿）。
- 解析器初始化 一个JSON解析器需要先构建字段映射再进行解析，这会花100ms到200ms，很明显的拖缓App启动时间。
- 垃圾回收 在解析JSON时创建了很多小对象，在我们的试验中，解析20kb的JSON流时，要分配大约100kb的瞬时存储，对Java内存回收造成很大压力。

FlatBuffers vs JSON

我尝试使用 FlatBuffers 和 JSON 解析 4mb 的 JSON 文件。

FlatBuffers 花了 1-5ms，JSON 花了大约 2000ms。在使用 FlatBuffers

期间 Android App 中没有 GC，而在使用 JSON 时发生了很多次 GC。在使用 JSON 时 UI 完全卡住，所以真实使用时只能在后台线程进行解析。（见左图）

如何使用 FlatBuffer 呢

我在我的 GitHub 中写了一个示例，里面手把手教你如何使用 FlatBuffer。

我为什么选择 Angular 2

作者 汪志成

没有选择是痛苦的，有太多的选择却更加痛苦。而后者正是目前前端领域的真实写照。新的框架层出不穷：它难吗？它写得快吗？可维护性怎样？运行性能如何？社区如何？前景怎样？好就业吗？好招人吗？组建团队容易吗？

每一个框架都得评估数不清的问题，直到耗光你的精力。这种困境，被称为“布利丹的驴子” —— 一只驴子站在两堆看似完全相同的干草堆中间，不知道如何选择，最终饿死了。

当然，那只是一个哲学寓言。现实中，大多数人采用了很简单的策略来破解它：跟风，选择目前最流行的那个。这是一个低成本高收益的策略，不过也有风险：成为现实版的《猴子下山》。最理想的方案还是要看出这两堆“干草”之间的差异，选择更适合自己的那个。

本文就将带你了解 Angular 2 这个“干草堆”的各种细节。

ALL-IN-ONE

不管是 1 还是 2，Angular 最显著的特征就是其整合性。它是由单一项目组常年开发维护的一体化框架，涵盖了 M、V、C/VM 等各个层面，不需要组合、评估其它技术就能完成大部分前端开发任务。这样可以有效降低决策成本，提高决策速度，对需要快速起步的团队是非常有帮助的。

让我们换一种问法吧：你想用乐高搭一个客厅，还是买宜家套装？

Angular 2 就是前端开发领域的“宜家套装”，它经过精心的前期设计，涵盖了开发中的各个层面，层与层之间都经过了精心调适，是一个“开箱即用”的框架。

当然，你可能还记着 Angular 1 带给你的那些快乐以及……痛苦。这是有历史原因的。由于它是从一个用来做原型的框架演化而来的，加上诞生时间很早（2009 年，作为对比，jQuery 诞生于 2006 年），当时生态不完善，连模块化机制都得自己实现（这就是 angular.module 的由来，也是 Angular 2 中抛弃它的理由）。在长达七年的演化过程中，各种进化的遗迹非常明显，留下了不少“阑尾”。

但 Angular 2 就不同了，它的起点更高，整合了现代前端的各种先进理念，在框架、文档、工具等各个层面提供了全方位的支持。比如它的“组件样式”能让你在无需了解 CSS Module 的前提下获得 CSS Module 的好处，它的 Starter 工程能让你在无需了解 Webpack 的前提下获得 Hot Module Replacement 等先进特性，它能让你从 Web Worker（你知道这是什么吗？）中获得显著的性能提升。

你只管在前台秀肌肉吧！剩下的，让 Angular 在幕后帮你搞定！

模块化的技术

虽然 Angular 是一个 ALL-IN-ONE 的框架，但这并不妨碍它同时是一

个灵活的框架。还记得 OCP（开闭原则）吗？一个好的设计，对扩展应该是开放的，对修改应该是关闭的。

Angular 2 很好的践行了 OCP 原则以及 SoC（关注点分离）原则。

它非常有效的分离了渲染与交互逻辑，这就让它可以很好的跟包括 React 在内的渲染引擎搭配使用，除此之外，它还可以使用内存渲染引擎，以实现服务端渲染；还可以使用 Native 渲染引擎，以编译出真正的原生程序（NativeScript）。

它还分离了数据供应与变更检测逻辑，从而让它可以自由使用包括 RxJS 以及 ImmutableJS 在内的第三方数据框架 / 工具。

不仅如此。

在 Angular 1 和 Angular 2 中最具特色的应该算是依赖注入 (DI) 系统了，它把后端开发中用来解决复杂问题、实现高弹性设计的 DI 技术引入了前端。Angular 2 中更是通过引入 TypeScript 赋予它更高的灵活性和便利性。

不过，Angular 2 并没有敝帚自珍，把它跟框架本身紧紧黏结在一起，而是把它设计成了一个独立可用的模块。这就意味着，无论你正在使用什么前端框架，甚至 NodeJS 后端框架，都可以自由使用它，并从中获益。

全生命周期支持

除非你打算写一个一次性应用，否则软件的生命周期会很长。宏观来看，真正的挑战来自多个方面，而且在不断变化。

开始的阶段，我们需要非常快速的建立原型，让它跑起来，引入最终用户来试用，这个时候，挑战来自开发速度以及可复用资产。

之后，会建立一个逐渐扩大的项目组，来完善这个原型的功能。主要

的挑战是让这个原型通过重构不断进行演化，特别是在演化的后半个阶段，产品需要保持随时可以上线的状态。但技术上的挑战那都不是事儿！关键是人。

如何让新人快速融入项目组，贡献生产力？这可没那么简单。“你先去学 xxx 0.5、yyy 0.9、zzz 5.3... 还要了解它们前后版本之间的差异，我再给你讲代码”，这种话，我可说不出口。一个优秀的框架需要对分工提供良好的支持，每个人都可以先从一些简单任务开始，逐步的从修改一个文件扩大到修改一个目录再到独立实现一个特性。

有了这种分工，每个团队成员就可以从一个成功走向一个更大的成功。这就需要框架在设计上具有良好的局部性：你可以放心大胆的修改一部分，而不用担心影响另一部分。你可以只修改 CSS，可以只修改 HTML，可以只修改 TS/JS，而不用担心自己的修改破坏了其他部分。而 Angular 2 通过声明式界面、组件样式以及独立模板文件等对这种安全感提供了有力的保障。

再然后，如果你的软件顺利的进入了线上维护阶段，那么恭喜你，你终于迎来真正的大 Boss 了！这个大 Boss 就是可维护性。Angular 开发组有很多程序员来自 Google，如果要问谁的软件维护经验最丰富，Google 和微软必然名列前茅。微软通过 TypeScript 的强类型机制体现了自己的经验，而 Google 则通过将 OCP、SoC、SRP 等一系列软件设计原则融入 Angular 体现了自己的经验。具体技术上则体现为：DI、生命周期钩子、组件等等。

最后，如果你的软件完成了历史使命需要逐步退出，是不是就能松口气了？不行，你还得继续留人维护它。如果你选择的是一个闭源的或某个社区很羸弱的开源技术，你就把以前的主力架构师、资深程序员留下来

继续维护它吧。或者你就得鼓起勇气跟用户说：你们玩儿，我先走了。而 Angular 是一个大型开源项目，并得到了 Google 的鼎力支持。即使经历过 Angular 2 项目组初期的公关失败，它仍然有着其它竞品无法企及的繁荣社区 —— 无论在全球还是在中国。显然，无论是对传统社区资源的继承，还是对新社区资源的开拓，我们都有理由相信，Angular 社区仍将一如既往地繁荣。至少，如果你不想让自己的软件建立在一大堆由个人维护的核心库上，那么 Angular 毫无疑问是最好的选择。

逃离“版本地狱”

如果是一两个人开发一个预期寿命不到一年的系统，那么任何框架都可以胜任。但，现实中我们都把这种系统称之为“坑”。作为一个高度专业、高度工程化的开发组，我们不能把“坑”留给友军。这些坑中，最明显的就是“版本地狱”。

想象一下，你仅仅升级了库的次版本号，原来的软件就不能用了，损坏了 N 处单元测试以及 M 个 E2E 场景。这种情况对于开发组来说简直是一个噩梦 —— 毕竟，谁也不想做无用功，更何况是一而再、再而三的。或者，它每次小的改动都会修改主版本号 —— 对，我就是不向后兼容，你能咋地？你所能做的就是每一次决定升级都如临大敌，不但要小心翼翼的升级这个库本身还要升级与它协作的几乎所有库。

可见，乱标版本号可能会让使用者付出巨大的代价。这不但体现在工作量上，还会体现在研发团队的招募与培养上，想象一下，对小版本之间都不兼容的框架，研发团队都需要记住多少东西？那简直是噩梦！

但是，版本号的问题在业内早就有了事实性标准，那就是 SemVer 语义化版本标准。唯一的问题是，作为框架 / 库的作者，遵循 SemVer 标准

需要付出的努力是巨大的。是否愿意付出这些代价，是一个框架（及其开发组）是否成熟的重要标志。

Angular 就是这样一个框架，其开发组曾作出的努力是有目共睹的。如果你是从 Angular 1.2 开始使用的，那么你为 1.2 所写的代码都可以毫无障碍的迁移到最新的 1.5 版，新的版本只是引入了新特性，而没有破坏老特性。这是因为 Angular 开发组写了大量单元测试和 E2E 测试，借助 CI 来保障这种平滑过渡。只有在从 1.0 到 1.2 的迁移过程中（1.1 一直是 beta，忽略不计），出现了一系列破坏性变更，这些变更被明确的记录在文档中，并且解释了做出这些变更的理由 —— 大多数是因为提升前端代码安全性。即使你恰好遇到了这些破坏性变更，它也会给出明确的错误提示。

这些必要而无聊的工作，正是帮助我们逃离“版本地狱”的关键。是否愿意做这些无聊而琐碎的工作，是一个框架的开发组是否成熟、专业的关键特征。而 Angular 的开发组已经证明了它值得你信任！

遇见未来的标准

编程领域，创新无处不在。但对一个工程团队来说，最难得的是标准。标准意味着：

- 招人容易。因为标准的东西会的人最多，而且人们愿意学它 —— 因为知道学了就不会浪费。
- 养人容易。因为网上有很多教程可用，即使不得已自己做教程，性价比也是最高的。
- 换人容易。标准就意味着不是私有技术，一个人离开了，就能用另一个人补上，而不会出现长期空缺，影响业务。

但是，标准永远会比创新慢一拍或 N 拍。这就意味着如果你追新，那么在获得技术上的收益的同时，也要付出工程上的代价。固然，两者不可兼得，但是还是有一些策略能够调和两者的。最简单的策略就是：遇见未来的标准。

所谓未来的标准，就是某个标准的草案，它很有希望成为未来的标准，这代表了业界对某项技术的认可，于是，即使它还不成熟，人们也愿意为之投资。比如虽然 Google 曾经提出过 N 种自有技术，包括 SPDY、Gears、OGG 等，但却并没有把它们变成私有技术，而是对社区开放以及并入 W3C 的标准草案。

Angular 2 采用的就是这种策略。它所参照的标准草案比较多，一个是 WebWorker，它借助 WebWorker 来把繁重的计算工作移入辅助线程，让界面线程不受影响；另一个是 WebComponents，Angular 2 的组件化体系就是对 WebComponents 的一种实现；最后是 CSS scoping，现在虽然市面上有了很多 CSS 模块化技术，但事实上最终还是会被统一到 CSS Scoping 标准上，届时，如果 :local 等关键字无法进入标准，就会成为私有技术。而 Angular 2 选择的方式是直接实现 CSS scoping 标准草案，比如 :host、:host-context 等。显然，采用这种策略，“遇见未来的标准”的成功率会高得多。

可以看到，Angular 2 的设计哲学中贯穿着对标准化的热烈拥抱，这是我判断它将赢得未来的另一个信心来源。

速度与激情

Angular 2 终于摆脱了旧的技术框架束缚，开始了对速度的极致追求。在 Angular 1 中，虽然绝大多数场景下性能都不是问题，不过还是因为其

代码中存在的一个用来实现脏检查的三重循环而饱受抨击 —— 似乎真有人能感受到 30 毫秒和 100 毫秒的差异似的。

不过，有软肋总是不太好。于是，在 Angular 2 中，通过重新设计和引入新技术，从原理上对速度进行了提升，据官方以前提供的一个数据说是把“变更检测”的效率提升了 500%。

它在“变更检测”算法上做了两项主要的改进。

在设计上，把以前的“多轮检查、直到稳定”策略改成了“一轮检查、直接稳定”策略。

当然，这会对自己的代码产生一定的限制，但实际上只在有限的少数几个场景下才会遇到这个限制，而且官方文档中也给出了明确的提示。

在实现上，把“变更检测”算法移入了由 WebWorker 创建的辅助线程中执行。

现代的家用电脑很多都已经是多核超线程的，但是浏览网页时实际上无法充分利用全部线程，而 WebWorker 提供了一个新的机制，

让一些相对繁重的计算工作运行在辅助线程中，等执行完了再通知主线程。即使你不明白 WebWorker 的工作原理，

至少也能知道 Ajax 请求是不会阻塞界面响应的，WebWorker 也一样。

除此之外，Angular 还对数据源进行了抽象，你完全可以用 ImmutableJS 来作为 Angular 的数据源，获得其在提升“变更检测”速度方面的所有优势。

除了“变更检测”外，Angular 以及所有前端 SPA 程序，都有一个通病：首次加载太慢，要下载完所有 js 和 css 才能渲染出完整的界面来。React 通过虚拟 DOM 解决了此问题，而 Angular 2 则通过独立的服务端渲染引擎解决了这个问题。我们前面说过，Angular 2 把渲染引擎独立了出来，因

此可以在 NodeJS 中实现服务端的内存渲染。所谓内存渲染就是在内存中把 DOM 结构渲染出来并发回给浏览器，这样，客户端不需要等 JS 代码下载完就可以显示出完整的界面了。这种分离还带来了另一个好处，那就是 SEO。SEO 同样是传统 SPA 的一个难点，它和服务端渲染是同一个问题的两个方面，因此随着服务端渲染的完成，SEO 问题也被顺便解决了。

让你无缝升级

Angular 2 开发组在早期阶段曾犯下一个严重的公关错误：过早宣布不兼容 Angular 1，也不提供从 Angular 1 到 2 的升级方案。

这让 Angular 2 开发组付出了沉重的代价，可谓伤透了粉丝的心。作为技术人员，这种失误固然是可以理解的，但却需要付出更多的努力来弥补它。而 Angular 2 确实这么做了。

在 Angular 2 中，开发组提供了一个 UpgradeAdapter 类，这个升级适配器让 Angular 1.x 的所有部件都能和 Angular 2.x 中的所有部件协同工作。

而最牛的地方在于，它让你可以一个文件一个文件的逐步升级到 Angular 2，而在整个升级过程中，应用可以继续在线上跑！看着神奇吗？其实也算不了啥，Google 做这种事早就是轻车熟路了。这只不过是对 Angular 开发组出色的工程化开发能力的一点小小证明而已。

不过，既然如此，当初又何必急匆匆宣布不兼容呢？可见，再出色的工程团队，如果缺少了和社区沟通的产品运营技巧，也会付出巨大代价。希望 Angular 开发组以后能谨言慎行，多用行动来平息质疑。

幕后 —— 当 Google 牵手微软

当年的浏览器大战，让人记忆犹新，Chrome 的出品商 Google 和 IE

的出品商微软正是浏览器大战的两大主角。

俗话说：没有永远的朋友，也没有永远的敌人，只有永远的…… 精益求精。

正是在这样的“俗话”指导下，Google 与微软相逢一笑泯恩仇，撤销了很多针对彼此的诉讼，甚至在一些领域开始合作。而实际上，在他们公开和解之前，就已经开始公开合作了，其契机就是 Angular 2。

这就要扯出一点小八卦了。

在 Angular 2 开发的早期阶段，由于传统 JS（ES5）以及当时的 ES6 草案无法满足项目组的开发需求，项目组有点烦。后来有人灵机一动开发了一种叫做 AtScript 的新语言，它是什么呢？一个带类型支持和 Annotation 支持的升级版 JS。其实在类型支持方面，TypeScript 早就已经做了，而且那时候已经比较成熟，唯一的遗憾是不支持 Annotation，也就是像 Java 中那样通过 @ 符号定义元数据的方式。

Angular 开发组就这样孤独的走过了一小段儿时间，后来也不知道怎么这两个欢喜冤家就公然牵手了。总之，最后的结果是：TypeScript 中加入了与 Annotation 相似的 Decorator 特性，而 Angular 放弃了 AtScript 改用 TypeScript。

这次结合无论对两大厂商还是对各自的粉丝，都是一个皆大欢喜的结局，称其为世纪联手也不为过。此后，Google 与微软不再止于暗送秋波，而是开始了一系列秀恩爱的动作。无论如何，对于开发者来说，这都是一个好消息。

软粉们可能还记得在 6.1 的微软开发者大会上，微软曾公开提及 Angular 2。事实上，TypeScript 目前虽然已经相当完备，但应用度仍然不高。就个人感觉来说，Angular 2 将是 TypeScript 的一个杀手级应用。

TypeScript 如果当选 TIOBE 年度语言，Angular 2 的推出功不可没。

为什么我要特意插播这段儿故事呢？那是因为我认为一个产品的未来最终取决于开发组的未来，而不是相反。软件的本质是人！一个心态开放、讲究合作、勇于打破陈规陋习的开发组，才是框架给人信心的根本保障。

后端程序员的终南捷径

前端程序员自不必说，因为有很多人就是靠 Angular 进入专业前端领域的。下面这段话主要写给后端程序员。

不管是想学习新技术还是出于工作需要，都有很多后端程序员对前端技术跃跃欲试。但面对前端让人眼花缭乱的大量候选技术以及未知的概念，他们往往感觉感觉无所适从。如果你是其中的一员，那么 Angular 可以“治愈”你的选择恐惧症。

正如前面所说，Angular 是一个“ALL-IN-ONE”的框架，这就意味着你只要掌握了 Angular 就可以完成大量的前端工作了。

这首先得益于它对各种技术的封装，让你不用关心它的实现细节。Angular 隔离了浏览器的细节，大多数工作你甚至都不需要知道 DOM 等前端知识就可以完成。

其次，Angular 选择了 TypeScript 作为主语言。如果你是个 C# 程序员，一定会对它的语法感觉似曾相识。没错，TypeScript 和 C#、Delphi 有同一个“爹” —— 传奇人物 Anders Hejlsberg。即使是 Java 程序员，也不会觉得陌生：强类型、类、接口、注解等等，无一不是后端程序员们耳熟能详的概念。说到底，并没有什么前端语言和后端语言，在语言领域耕耘多年的 Anders 太熟悉优秀语言的共性了，他所做的取舍值得你信赖。

再次，Angular 在前端实现了服务与依赖注入的概念。随着前端需求

的进一步膨胀，即使只算逻辑代码，其需求复杂度也即将甚至已经赶上了大部分后端程序。所以，后端遇到过的挑战前端也迟早会遇到，这正是后端程序员弯道超车的好机会，而依赖注入正是后端程序员的看家本领。

最后，Angular 对团队作战提供了良好的支持，比如模板与代码的分离、样式表的局部化、组件化的设计、服务与依赖注入体系等。这些特性让后端程序员可以先专注于跟后端代码最像的模型和交互逻辑部分，而把诸如样式、模板等自己不擅长的部分交给队友。

IBM、Google、Oracle 三巨头的公有云之殇（上）

作者 刘黎明

编者按

InfoQ 中文站新推出《洞见云计算》专栏，精选包括刘黎明个人公众号上的文章，让更多的读者朋友受益，本栏目的内容都经过原作者授权。本文是《云计算演义》系列文章第三篇。

企业 IT 之王 IBM，百年老店，吃的盐比别人吃的饭多。互联网之王 Google，躺着赚钱，技术和创新能力人所共知。数据库之王 Oracle 在企业核心业务中，如泰山般不可撼动。他们在各自的壕沟里，都是无人可以挑战的王者。现在，他们的共同点是：狂奔在与关闭公有云赛跑的路上。

原本是要一个一个公司说的，三个一起说多少还是有些别扭，但是为什么还是要一起说呢。他们虽然出身各不相同，现在和将来在云计算上的策略也各不相同，但在他们的云计算当前的境况和将来的走势，可能在轨迹上，有相似之处。而且，他们现在同处第二集团。

我并不准备探讨这些公司的所有业务，包括其他重要业务，我们的关注点肯定在云计算业务。但我们也必须在讨论云计算业务之前，对这些公司的历史有一个概要了解，我们将更容易理解这些公司的云计算路径。参军也好，参加工作也好，都还要做背景调查呢，何况我们谈一个公司的云计算业务。

IBM 极简史

当然没有人不知道 IBM，IBM 的历史确并不是很多人知晓，但 IBM 的历史并不是无足轻重。如果你十分熟悉 IBM 历史，跳过。

1874 年 2 月 17 日，IBM 创始人 Thomas J. Watson 出生，但这位现在被称为老 Watson 的人并不是 IBM 严格意义上的创始人。

1896 年，Herman Hollerith 在华盛顿特区成立制表机器公司（Tabulating Machine Company），这是世界上第一家电子制表与财务审计机器公司。

1901 年，George W. Fairchild 成立国际时间记录公司（International Time Recording Company），该公司是邦迪制造公司、威拉德与弗里克制造公司和标准时间印章公司三家公司的销售公司，同时还生产卡片记录器。

1907 年，国际时间记录公司收购戴伊时间记录器公司（即 1883 年成立的戴伊专利权公司），该公司生产刻度盘、卡片和工时记录器。

1911 年，Charles R. Flint 筹划了国际时间记录公司、计算尺公司和制表机器公司三家公司的合并，成立了计算－制表－记录公司（Computing-Tabulating-Recording Company，即 C-T-R 公司）。George Fairchild 成为公司的董事会董事长。

1914 年 5 月 1 日，年轻销售员 Thomas Watson 被 CTR 录用，后来他成为蓝色巨人 IBM 的缔造者。

1915 年，Thomas Watson 当选 CTR 公司总裁兼总经理。

917 年，CTR 公司以国际商用机器有限责任公司（International Business Machines Co., Limited，IBM）的名字进入加拿大市场。

1924 年，计算 - 制表 - 记录公司（即 C-T-R 公司）改名为国际商用机器公司（IBM 公司）。四分之一世纪俱乐部（Quarter Century Club）成立，该俱乐部只承认那些为公司工作 25 年以上的员工。

二次世界大战期间，IBM 进入计算领域，1944 年，IBM 公司向哈佛大学赠送其首台大型计算机 -- 自动顺序控制计算机，也被称为 Mark I。

1951 年，IBM 开始决定开发商用电脑，聘请冯 • 诺依曼担任公司的科学顾问，1952 年 12 月研制出 IBM 第一台存储程序计算机，也是通常意义上的电脑，IBM 701。这是 IT 历史上一个重要的里程碑。

1952 年，小沃森出任 CEO，IBM 新一代领导集体诞生。

1964 年 4 月 7 日，IBM 主席 Tom Watson 亲自发布 System 360。IBM 公司在 1967 年控制了大型机市场的 76%。

1971 年，IBM 公司的计算机引导 " 阿波罗 14 号 " 和 " 阿波罗 15 号 " 宇宙飞船成功登月。

1975 年，IBM 推出首款型号为 5100 的“便携式”计算机。它重 50 磅（约合 23 千克）。根据存储空间大小（16KB、64KB）和选购存储配件（如 8 英寸的软盘）的不同，售价从 9000 到 20000 美元不等。

1981 年 8 月 12 日，IBM 发布第一台 PC，Intel 和微软的开始萌芽。在此之前 1976 年，苹果已经发布了 Apple I。晚了四年，但是已经推出抢去 Apple 四分之三的市场。

1984 年 3 月 1 日 Commodore 公司宣布将生产 IBM PC 兼容机，当时 IBM 和苹果各得 28% 和 25% 的市场份额。

1986 年 4 月 3 日，IBM 推出第一台公文包大小的膝上型（laptop）电脑。

1987 年 1 月 20 日，DEC 推出超级计算机挑战 IBM。

1987 年 12 月 4 日，IBM 发布多任务操作系统 OS/2 第一版，但是后来

败于微软之下。

1990 年 8 月 1 日，IBM 出售打字机和键盘业务。

1991 年 2 月 28 日，IBM 宣布裁员 10000 人。

1991 年 4 月 12 日，IBM 和苹果召开秘密会议，商讨推 PowerPC，对抗 Wintel。

1991 年 7 月 3 日，IBM 和苹果宣布结盟，共同开发应用于 RISC 芯片的操作系统。

1991 年 10 月 2 日，IBM 和苹果宣布建立合资公司，开发和销售 PowerMac。

1992 年 8 月 22 日，IBM 宣布推出个人数字助理 PDA。

1992 年 12 月 16 日，IBM 宣布 50 来的第一次大裁员。

1993 年 1 月 20 日，IBM 宣布历史上的第一次亏损。

1993 年 3 月 26 日，Lou Gerstner 被任命为 IBM 公司新 CEO，拯救危难中的蓝色巨人。在接下来的十年间，IBM 剥离了利润较低的业务，如 DRAM、网络、个人打印机和硬盘等，并大力投资软件和服务，收购 Lotus 软件和普华永道。

1994 年 5 月 25 日，Compaq 超过 IBM 和苹果，成为全球 PC 之王。

1994 年 8 月 17 日，IBM 宣布推出基于计算机网络的电子定购系统。

1995 年 1 月 12 日，IBM 在专利竞赛中获得桂冠，专利数量名列美国公司第一。

1995 年 2 月 3 日，IBM 宣布员工上班时间可以穿便服，一改传统。

1995 年 6 月 11 日，IBM 宣布巨资购并 Lotus 软件公司。Lotus 此时已经败于微软之下。

1996 年 12 月 16 日，IBM 宣布了 PowerPC 的死刑。

1997 年 5 月 11 日，IBM 的“深蓝”（Deep Blue）计算机击败世界象棋大师 Gary Kasparov。

1997 年 6 月 10 日，IBM 宣布将关闭在线购物网站 WorldAvenue。

2002 年，IBM 的研发人员共累积荣获专利 22358 项，这一记录史无前例，远远超过 IT 界排名前十一大美国企业所取得的专利总和，这 11 家 IT 强手包括：惠普、英特尔、Sun、微软、戴尔等。IBM 在 2012 年获得了 6478 项美国专利，刷新该公司的历史新记录。

2005 年，IBM 将 PC 业务出售给联想。

2006 年，全年经营业绩——收入总额 914 亿美元。

2009 年，IBM 总营收 1036 亿美元，全球雇员约 398455 人。

2011 年，IBM 以创办人老沃森为名，推出史上第一台听懂人类自然语言的超级计算机（沃森）。

2011 年 2 月 17 日，IBM 的超级计算机“沃森”（Watson），在美国老牌知识问答电视节目——“危险边缘”（Jeopardy!）击败了两位人类冠军，被誉为 21 世纪计算机科学和人工智能方面的伟大突破。沃森涵盖了大约 2 亿页的内容（约一百万册书籍的价值）。

2011 年 9 月 30 日，截至周四收盘，IBM 的市值达到 2140 亿美元，从而超越微软成为全球市值第二高的科技企业。这是 15 年来 IBM 市值首超微软。

2012 年，IBM 成为美国雇员最多的公司，全球拥有 345000 名员工。

2014 年 8 月，IBM 以 23 亿美元的价格，将低端服务器业务出售给联想。

这么啰嗦，还叫极简史？没错，这就是 IBM。IBM 历史悠久，最重要的是，IT 史短短半个多世纪的历史上，IBM 无论是从技术还是商业上，都引领 IT 行业的创新和潮流。每隔一段时间，IBM 就能推出最重要的技术、

产品，并给 IT 行业带来巨大的影响。

IBM 在材料、化学、物理等基础科学领域也有很深造诣。硬盘、扫描隧道显微镜（STM）、铜布线技术、原子蚀刻技术都是 IBM 研究院发明。

IBM 的一篇论文就能诞生一个大业务，比如关系数据库，一个产品两个合作就能诞生几个巨头，比如微软和 Intel。某种程度上说，Google 如今只是在某些领域做到了前者，后者却远远不及。

IBM 今天仍然是对企业最重要的 IT 公司，没有之一。IBM 是当今世界上第二大软件公司、第二大数据库公司、第二大服务器公司、第三大安全软件公司、第六大咨询公司，连续 14 年是最大的应用基础设施和中间件公司，占有约百分之三十的市场份额。

大部分现在风头正劲的公司，50 年后是否存在还是个问号。但 IBM 见证了 DEC、Unisys、Compaq 的起起落落，见证了数次 IT 变革和公司自身的转型。从历史惯性来说，它不会失败。

IBM 当然从未在消费者市场成功过，但在半个多世纪的企业市场和政府市场里一直是 IT 老大，所以他给人的一直是信任和可靠。连从不投资科技和 IT 公司的股神巴菲特，也对 IBM 股票情有独钟。

IBM 开创并一直统治大型机时代，开创了 PC 时代，在互联网时代仍然是重要的背后玩家，但云时代，我们今天看到了它的努力，但也有很多可以批评。

Oracle 极简史

Oracle 的重要性无需赘言，它可以称为 IT 软件领域躺着挣钱的公司。Oracle 的历史也有三十多年，而且其起步和发展均与 IBM 密切相关。

1970 年的 6 月，IBM 公司的研究员埃德加·考特（Edgar Frank

Codd）在 Communications of ACM 上发表了那篇著名的《大型共享数据库数据的关系模型》（ARelational Model of Data for Large Shared Data Banks）的论文。这是数据库发展史上的一个转折点。当时，层次模型和网状模型的数据库产品在市场上占主要位置。从这篇论文开始，才拉开了关系型数据库（RDBMS）软件革命的序幕。

IBM 虽然 1973 年就启动了 SystemR 的项目来研究关系型数据库的实际可行性，但没有及时推出这样的产品，因为当时 IBM 的的 IMS（著名的层次型数据库）市场销路不错。首先，如果推出关系型数据库，牵涉到 IBM 很多人的自身利益。其次，没有人有强大的动力搞一个新型的数据库。要发财，卖好已有的广受欢迎的产品就好了。

1977 年 6 月，LarryEllison 与 Bob Miner 和 Ed Oates 在硅谷共同创办了一家名为软件开发实验室（SoftwareDevelopment Laboratories，SDL）的计算机公司（ORACLE 公司的前身）。

1978 年，公司迁往硅谷，更名为“关系式软件公司”（RSI）。

1982 年，RSI 更名为 Oracle 系统公司（Oracle System Corporation），Oracle 公司。用产品名称为公司命名，帮助公司赢得了业界的认同。

1983 年 3 月，RSI 发布了 ORACLE 第三版。从这个版本开始，ORACLE 产品有了一个关键的特性：可移植性。

1983 年 Oracle 决定开发便携式 RDBMS。Oracle 开发出 V3，这是第一款在 PC 机、小型机及大型机上运行的便携式数据库。

如果你要了解详细的 Oracle 发展历史，参考 Fenng 的《书写历史的甲骨文 --ORACLE 公司传奇》。

如果你把 Oracle 当成一个数据库公司，Oracle 的历史从这里就没有

什么新东西了。它只是不断开拓新市场，打败挑战者如 IBM 和 Sybase，发布新的数据库版本。从主机时代，到客户端 / 服务器架构时代，到 Internet 时代，Oracle 的数据库产品都紧跟潮流。

到 1995 年，Oracle 营收 29.67 亿美元。拉里 • 艾利森发布网络计算机的概念：网络计算机是一种带有本地内存的互联网应用设备，一种不含硬盘驱动器的快速微处理器。

2004 年 12 月 13 日，Oracle 公司宣布签订了以每股 26.50 美元、总计约 103 亿美元的代价收购仁科 (PeopleSoft)。仁科当时是 ERP 领域的老二，老大是 SAP。从此，Oracle 在企业应用软件市场上一跃成为第二名。

从 2005 年开始，Oracle 的发展历史差不多就是其收购史，陆续收购了 DEC 的 Rdb、Innobase、Siebel、BEA、Hyperion、SUN。对，Innobase 就是开发 MySQL 存储引擎 InnoDB 的公司。SUN，没错，就是那个鼎鼎大名的 SUN。

在收购战略之前，Oracle 是第一大数据库软件公司。在收购 SUN 之前，Oracle 已经凭借收购成为重量级的应用基础设施和中间件公司，在中间件、ERP、CRM、商业智能市场都能拍到第二或前几名。此时已经是第二大应用基础设施和中间件公司。

在收购 SUN 之后，Oracle 已经成为最重要企业 IT 全栈公司之一，另一家是 IBM。也就是说几乎什么都做，而且很多还做得不错。

Google 极简史

Google 等于搜索，这样说没有错。搜索等于 Google，这么说也错不到那里去。

1995 年，合伙人 Larry Page 和 Sergey Brin 在斯坦福大学计算机博

士候选人的春季聚会上见面，Brin 带 Page 转校园，两人相见恨晚。

1996 年，两人合伙做了一个搜索引擎叫 BackRub。由于它太消耗带宽，斯坦福大学最终把它从学校的服务器上撤了下来。

1997 年，搜索引擎的名字变为 Google，来自单词“googol”（意思是 10 的 100 次方）。象征着他们想整合容量无限扩大的网络世界的野心。

1998 年，当 Sun 公司创办人 Andy Bechtolsheim 准备给谷歌 10 万美金的投资时，Page 和 Brin 意识到他们应该自己创业，于是当年 9 月 4 日于加利福尼亚注册了属于他们自己的公司。在 9 月 7 日组建公司，位于加州一个车库，有 4 名员工。布林和佩奇从家人、朋友和投资者募集了 100 万美元。当时布林和佩奇分别为 24 岁和 25 岁。成立数天后，公司注册了 Google.com 域名。

1999 年 2 月，他们的 8 名员工在帕洛阿尔托有了新办公室。在谷歌第一版的发布后，使它们获得了 Sequoia Capital 和 Kleiner Perkins 两千五百万美元的投资。

2000 年，谷歌与 350 个客户发布关键字广告销售系统 Adwords。Google 每天进行 1800 万次查询，成为最大的互联网搜索引擎，雅虎选择 Google 作为默认的搜索结果供应商。年底时，提供 15 种语言的搜索服务。Google 第一笔融资花到一半的时候，上市了。

2001 年，曾任过 Sun 公司首席技术执行官和软件公司 Novell 首席执行官的 EricSchmidt 出任谷歌公司的首席执行官和董事会主席。图片搜索在 7 月上线。

2003 年，美国方言协会选取“Google”为 2002 年最有用的单词。1 月，Google 收购了 Pyra 实验室，这是网络出版工具 Blogger 的创建者。5 月，Google 推出 AdSense，这一广告计划能按照网站内容做广告。

2004 年，以 85 美元每股的发行价，集资 1960 万美元。上市让公司搬到了新总部——可以容下超过 800 人校园般的“Googleplex”。2 月，雅虎开始推出自己的搜索技术，淡出 Google 搜索技术。3 月 31 日：Google 宣布了免费电子邮件服务 Gmail。8 月收购了数字绘图公司 Keyhole，这成为了来年 2 月谷歌地球发布的契机。8 月 19 日，Google 在纳斯达克故事以 100.01 美元开盘。

2006 年 10 月，Google 公司以 16.5 亿美元，收购影音内容分享网站 YouTube，是 Google 有史以来最大笔的并购。

2007 年，股价首次超过 700 美元。“Google”以动词进入牛津英语词典。Gmail 开放使用（因为先前必须有邀请码才可以）。

2007 年 11 月 05 日，Google 宣布基于 Linux 平台的开源手机操作系统的名称为 Android。

2008 年，结合街景视图的新版本谷歌地球发布。谷歌浏览器 Chrome 提早发布。2008 年 9 月 7 日，Google Map 卫星升空，将为 Google Earth 提供 50 厘米分辨率高清照片。

2009 年，谷歌翻译支持 51 种语言，包括斯瓦西里语、威尔士语、意第绪语，相当于 2550 种语言对。道路导航系统发布，它可以使卡车司机仅仅使用手机就可以跑遍世界。

2012 年 5 月 21 日，Google 收购 Motorola 获中国政府批准，完成了 125 亿美元收购摩托罗拉移动。

2012 年 6 月 28 日，Google I/O 开发者大会在美国旧金山开幕。发布谷歌首款自主品牌平板电脑 Nexus7、外形前卫的社交流媒体播放器 NexusQ 以及酷炫的概念智能眼镜“谷歌眼镜”。

2012 年 10 月 2 日，谷歌已经超越微软，成为按市值计算的全球第二

大科技公司，原因是通过互联网进行的计算已经降低了台式机软件的市场需求。

其实从 2000 年，Gogole 发布 Adwords 以来，从商业上说，Google 再也没有大的变化了。它现在固然有很多的创新，包括 IT 技术的创新，但它的业务收入和利润基本上全部来自于搜索广告。

最近几年，人工智能（包括 2016 年初的围棋程序 Deepmind）、自动驾驶、热气球无线网络 ProjectLoon、手势操作技术 ProjectSoli、仿人机器人 Atlas、量子计算机等，Google 新的技术和商业项目走在世界前列。

如今 Google 不仅是互联网之王和搜索霸主，Google 在开源、技术创新、商业创新上的影响力，在 IT 界无人能及。

IBM 依旧是大象，尚能舞否，善于跳舞的大象遇到了危难

IBM 前 CEO 郭士纳在 2003 年出版了自传《谁说大象不能跳舞》，书中说“‘如果大象能够跳舞，那么蚂蚁就必须离开舞台”。从 1993 年到 2002 年，郭士纳帮助 IBM 的股价翻了约 9 倍。虽然营收增长了近 50%，考虑到 IBM 的体量，这仍然不容易，且让 IBM 回到了增长和盈利的轨道上来。

在 2004 年，联想收购 IBM 个人电脑事业部后，沃顿商学院管理教授、曾经在 IBM 作为工程师工作过 9 年的兹巴拉基（MarkJ.Zbaracki）的评论：″总的说来，这起交易是 IBM 非常具有弹性的又一个例子。IBM 的长处一直就是善于彻底改造自己。″

那时的 IBM，初步完成了从硬件向软件和服务的转型，是众多 IT 公司争相学习的榜样。不管是不是 IBM 的客户，都将 IBM 视为导师，而有合作机会时，选择与 IBM 合作是他们的首选。比如，华为的研发和管理体系就是以 IBM 为榜样，在 IBM 的亲手操刀下发展起来的。

如果从盈利的角度来看，亚马逊至今仍然是蚂蚁，近二十年来大部分的季度都在盈亏线上挣扎。在 AWS 上线三年后，亚马逊仍然可以认为是蚂蚁，认为那时没有任何迹象表明 AWS 能进入企业级 IT 市场。但是，《谁说大象不能跳舞》出版十年后的 2013 年，IBM 的舞步似乎突然沉重起来，亚马逊 AWS 击败 IBM，夺走中情局 6 亿美元云计算订单。如果一个单你觉得不重要，那么包括三星、辉瑞、PBS（公共电视网）和国家航空航天局、GE、Netflix 这些行业巨头都拥抱 AWS，还有什么对 IBM 是重要的呢？

从产品的性能和可靠性看，低端且不稳定的 X86 架构服务器也是是蚂蚁。但在 2012 年 X86 服务器占据 98% 的服务器总出货量。根据 IDC 的数据，在 X86 架构服务器的冲击下，IBM 基于 Power 架构服务器的收入在 2013 年第 2 季度同比下滑 25%。

IBM 一直在吃 90 年代以前的老本。IBM 凭借在大型机的统治地位，在政府和大型企业中建立了良好的信誉和客户关系。也正是基于此，在小型机夺得领先地位，并在 PC 市场占据了一席之地，尽管在 PC 时代 IBM 并不是领袖型厂商。

在互联网时代，IBM 更是藏于幕后，为运营商兴建数据中心出谋划策，为企业构建电子商务体系提供软件和服务。

可以说，在大型机时代之后，IBM 离最终用户越来越远，而 IT 离普通用户却越来越近。IBM 利用老本固守在政府和企业的 IT 圈里，这个圈也不断被 CA、HP、DELL、微软虎视眈眈并蚕食。

因此 90 年代初之后，尽管 IBM 的利润和营收、股价都有不俗的表现，但它在企业 IT 和整个信息行业的地位，越来越低了（见图 1）。

截至 2015 年第四季度，IBM 已经连续 15 个季度营收同比下滑。事实上，最近 8 年，IBM 一半以上的季度，营收是同比下滑的。

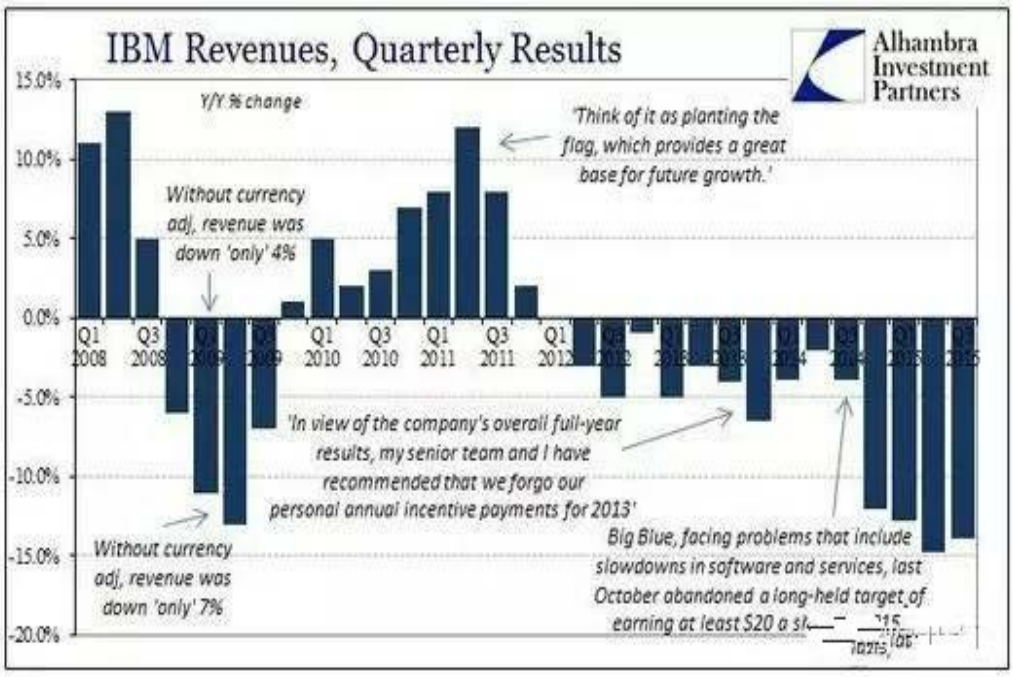


图 1

IBM 2008 年营收 1028.2 亿美元，2015 年 IBM 的营收 817.4 亿美元，总营收下滑了 20%，而同期全球 IT 服务市场增长了 20%。

1998 年 IBM 的营收已经达到 816.6 亿美元，1999 年 875.4 亿美元，其历史最高营收在 2011 年达 1069.3 亿美元。

也就是说 16 年过去了，IBM 的营收回到了起点。四分之一个世纪过去了，其营收增长不到百分之十五。考虑到货币贬值，16 年间其实际营收下滑了 50% 以上。

而同期的其他发展强劲的 IT 公司，这 16 年间，营收增长 10 倍以上，特别是 PC 时代和互联网前期的骄子微软、因特尔、Oracle、思科等。

这 16 年间，不要说消费者 IT 市场，即使企业 IT 市场也增长了 5 倍有余，而 IBM 的营收增长为零。

尽管企业在 IT 发展的任何阶段，都需要 IBM 这样的公司的帮助。但

进入二十世纪后，IT 设备和技术更新快、普及快，IBM 都难以跟上。加上设备的模块化、通用化，软件的开源化，一直到最后云计算的兴起，企业对 IBM 这样的公司的依赖性越来越低了。

现在，已经有了 All-In-Cloud 的中小企业甚至大型公司，那么，IBM 的日子几乎一眼就能看到头了。

20 年来，IBM 的口号从电子商务到随需应变，再到智慧地球。IBM 提出的每一个概念都成功预言了 IT 产业未来 10 年的变革方向。但是，每次都在坚持了数年的产业趋势即将爆发的时刻，IBM 却将机会留给了蚂蚁。当 IBM 在 2004 年用随需应变替代电子商务的品牌形象时，电子商务即将迎来爆发；当 IBM 在 2008 年转向智慧地球时，真正做到随需应变的云计算也即将大行其道。在 IT 产业，IBM 一直有着其他公司望尘莫及的战略视野和战略能力，但是，这头大象的舞步却越来越沉重。

情况很危急，罗睿兰面临的转型压力，绝对大于郭士纳。IBM 在云计算兴起之时，并没有 PC 兴起或互联网兴起时那么厚的底子，也没有那么大的行业影响力了。甚至人才资源、技术积累、管理能力都已经没有底子了。20 年前的 IBM，企业还离不开它，而今后，企业可能不在需要 IBM。

但是，但是，IBM 是会跳舞的大象。虽然，这一回，这舞步太沉重了。二十五年前的经验，十五年前的经验，放到今天，未必适用了。

从硬件到软件和服务，到云计算、商业分析、媒体和广告：IBM 的公司战略

2016 年 1 月，IBM CEO 第一次在全球消费电子展（CES）上做了演讲时，说 IBM 不再是一家硬件或软件公司，而是一家“认知解决方案云平台”公司。

看起来很玄乎是吧？但是就算不明白“认知解决方案云平台”是什么，

我们也知道，IBM 的视野已经小到一种业务场景的解决方案了，虽然后面带个云平台，那也只是为了把人知解决方案打肿脸充胖子。

20 世纪 90 年代之前，IBM 是一家大型机、小型机、PC 全系列的硬件、软件、解决方案的综合 IT 服务商，几乎能满足企业和个人的所有 IT 需求。

21 世纪之后，在高利润率的幌子下，卖掉 PC、X86 服务器后，在硬件领域的话语权基本只剩下开山老本行大型机。

IBM 说自己是“认知解决方案云平台”公司，当然不是说 IBM 只做认知解决方案，而是强调它在认知解决方案市场的地位。因为它仍是企业 IT 堆栈上，产品线最丰富的公司，能勉强与其相比的，只有 Oracle。

但，IBM 的视野仍然小了下去。尽管“认知解决方案云平台”是制高点，是高科技，甚至是未来的趋势，尽管这个说法是为了体现 IBM 云平台的与众不同之处：商业分析。没错，商业分析师 IBM 干云之前攒下的老底，这回云计算混战终于可以用上了。

自 2005 年以来，IBM 花 140 亿美元陆续收购了 120 多家提供各种“商业分析”服务的公司。无论是商业分析还是云计算，IBM 都大搞收购。这些收购集中在：

云计算

2013 年 6 月，收购 Softlayer，略低于 20 亿美元。

2015 年 6 月 3 日， IBM 宣布收购云解决方案供应商 Blue Box，Blue Box 为企业提供基于 OpenStack 的托管式私有云解决方案。

2015 年 7 月 23 日收购数据库管理工具 Compose。

2015 年 9 月 10 日收购 Nodejs 全栈框架 StrongLoop。

2015 年 10 月 5 日收购海量数据存储 Cleversafe 。

2015 年 11 月 3 日收购混合云管理服务软件 Gravitant。

商业分析

2015 年 3 月，IBM 收购 AlchemyAPI，平台上有近 4 万名开发者利用机器学习算法，做自然语言处理和图片识别。同月，IBM 收购搜索引擎 Blekko，Blekko 的 slashtag 功能能够过滤垃圾网页，并能人为优化搜索结果，用户用得越多，越能看到自己想要搜索的结果。

2015 年 4 月，IBM 成立 Watson Health 部门。8 月， IBM 10 亿美元收购医学成像设备提供商 Merge Healthcare。这能让沃森就看懂医学图像，包括核磁共振扫描图、X 光照片等。

2015 年 9 月，IBM 收购检测冒用信用卡行为的公司 IRIS Analytics。

2015 年 10 月，IBM 则花 20 亿美元收购了 The Weather Company 的数据。这些天气信息能够影响作物生长、物流、工业生产等。

2016 年 1 月，IBM 收购 IBM 收购 BI 软件厂商 Varicent。

媒体和广告

2015 年 12 月，收购 Clearleap。Clearleap 的技术确保用户在任何时间和设备上都能访问想看的视频。HBO、Verizon 以及美国橄榄球大联盟（NFL）都是它的用户。

2016 年 1 月，IBM 1.3 亿美元收下视频直播服务 Ustream，并成立一个新部门，专门处理媒体技术。

2016 年 2 月之前，IBM 收购了了 3 家数字营销公司，都放进了 IBM 互动体验（Interactive Experience ）部门。

如果抽象地说，IBM 自 90 年代初以来，在战略上一直有 2 个原则：专注高价值业务、去硬件化。

在郭士纳的后继者彭明盛担任 IBM CEO 的近 10 年间，虽然在营收上未有太大增长，却让 IBM 连续保持了双位数的利润增长。2003 年，IBM 总营收为 891 亿美元，利润 75.8 亿美元，全球雇员 23 万人，到彭明盛卸任 CEO 的 2011 年，IBM 的营收只增长了将近 20%（达 1069 亿美元），利润却增长 115%（达 163 亿美元），员工数几乎翻番（近 42.7 万）。利润增长是华尔街喜欢的，所以股价也从 2003 年不到 90 美元增长到 2011 年的 160 美元。

彭明盛是如何的呢？盯住运营优化和 EPS（每股盈余）！没有营收增长，就变卖低利润业务，降低业务成本。运营优化是这样的：采用全球共享服务削减通用的重复作业和额外管理费用，实现规模经济效应；在全球最佳地点运营，有效利用全球资源和能力；提高管理效率，通过减少内耗提高利润率。处于战略最顶层的是 EPS（每股盈余），然后通过 EPS 目标生成战略规划，再分解成业务运营计划和预算，最后通过层层分解形成各个区域、业务单元和行业的绩效指标。

这是管理效率和财务报表的优化，不是产品优化，不是产品创新。整个公司围绕着这个转。

在郭士纳时代，被众多企业争相学习的将 10% 的营收投入研发，在彭明盛时代早已变成了 6%。罗睿兰时代，这个 6% 怕是都要削减了。

最近 15 年来，IBM 维持增长的只有利润，这可能就是所谓的运营优化。而这基本靠工作向发展中国家转移，和人均成本的降低来建立的，基本上 IBM 的收入在知名 IT 公司里是偏低的。

这个战略，甚至可以很好地用来解释此前出售一些列业务，甚至 2009 年以来 IBM 至少出售了 10 宗业务。这在某种程度上也能解释十多年来 IBM 的营收没有增幅。

但这些战略和出售，并不能解释 IBM 在已经保有的核心业务上的营收下滑和市场份额的减少。所以，无论 IBM 自己还是媒体，用 IBM 的战略选择来解释营收下滑是不合理的。

我认为 IBM 近 10 年的核心战略是商业智能。

回到 2007 年 11 月 13 日，IBM 宣布将以 50 亿美元现金收购 Cognos，这是 IBM 有史以来单笔金额最大的收购活动，就是为了强化 IBM 在商业智能软件领域的竞争力。那一年，SAP 以 70 亿美元拿下了 Business Objects SA，甲骨文以 33 亿美元将 Hyperion Solutions Corp 收至麾下。可以说，商业智能是那些年的绝对热点，被寄予厚望。

那时，亚马逊的云计算服务，AWS EC2 和 S3 已经上线一年有余，即将进入第一个快速发展期，这些服务并没有进入 IBM、Oracle、Google、微软的视野，因为 IBM、微软、Oracle 眼里那时只有企业用户，Google 眼里虽有开发者，但绝没有经营基础设施生意的想法。而 AWS 那时真的只是个玩具，极少数开发者的玩具。

事实上，IBM 在最近 10 年至少收购了 40 家数据分析公司。2006 年 8 月，收购业务流程和内容管理软件公司 FileNet；2006 年 8 月，收购业务流程和内容管理软件公司 FileNet，SPSS 成为国际公认且标准的统计分析软件；2010 年 8 月，收购互联网商业分析软件商 Coremetrics；2011 年 9 月，收购智能与调查管理软件开发商 i2；2011 年 10 月，收购风险分析软件公司 Algorithmics；2012 年 2 月，IBM 收购基于云计算的数据分析软件的公司 Emptoris；2012 年 4 月，收购分析软件公司 Varicent；2013 年 2 月，收购非结构化企业数据分析及管理公司 StoredIQ；2013 年 2 月，收购非结构化企业数据分析及管理公司 StoredIQ。

从 2001 年至今，IBM 收购了 120 多家公司，其中 2010 年以前并购的

80 多家公司大多数都充实到以 Websphere、IM（BI）、Tivoli、Rational 和 Lotus 为主的品牌软件中。而 2010 年后 IBM 花费超过过 120 亿美元并购 40 多家公司的收购，分布在 6 个领域：云计算、企业管理、商业智能和数据分析、服务器和网络存储优化、企业治理合规与安全、人工智能。

除了研究和开发的投资，IBM 总计在大数据和数据分析这个领域投资已经超过 170 亿美元并完成 30 个以上的收购。

而在云计算领域的收购，基本都还是可以归纳到企业数据中心管理和企业管理软件中。自 2007 年开始，IBM 已先后收购了超过 15 家拥有云计算专业知识的技术企业。2010 年 5 月收购提供在线托管的软件程序服务 Cast Iron Systems；2012 年 2 月收购以色列移动应用开发和基础软件提供商 WorkLight；2013 年 11 月收购基于云的消息工具提供商 Xtify；2013 年 11 月 14 日，IBM 宣布收购以 MaaS360 的品牌提供基于云的企业移动管理解决方案公司 Fiberlink；2014 年 2 月 25 日宣布收购 CouchDB 项目的积极贡献者 Cloudant。

能为公有云提供帮助的公司，大概只有 Softlayer 一家。

如果收购什么，代表 IBM 看好什么，那么云计算远在 IBM 的战略雷达之外。我们不要看它怎么说，看它怎么做。IBM 看好的依然是：企业数据中心软硬件、企业管理软件，人工智能和商业分析。可以说，IBM 认为前两者是过去和现在，后者是将来。

IBM 的云计算，新瓶装商业分析的旧酒

外界和 IBM 自己都将 IBM 的业务大致划分为硬件、软件、服务。实际情况是，IBM 的软件增长乏力，硬件和服务持续下滑。IBM 靠在这三个方面一直靠收购解决营收和利润问题，而云计算持续，难以入其法眼。在

2013 年初，在 IBM 主办的一个 IDC 行业会议上，我说 IBM 的硬件、软件、服务正全面遭受云计算的侵蚀，可能有灭顶之灾。彼时，虽然云计算大势已经很明显，IBM 的对危机和将来仍然认识不足。

2013 年初那时，IBM 在公有云上还是零蛋。在 2013 年的档口，亚马逊 AWS 在 2011 年的营收即超过 10 亿美元，2012 年超过 17 亿美元，2013 年将达到 30 亿美元。Google 和微软已经有了各自的公有云产品，IBM 那时的公有云还是零蛋。IBM 的定力之强不得不佩服。

并购显然是 IBM 一贯的战略，虽然效果并不怎么样。嘘，IBM 已经失去了自我造血、创造新业务的基因和能力。云计算也是如此。

2013 年 6 月，IBM 终于收购了一家号称做云计算的公司 Softlayer，代价 20 亿美元。再这之前的 2013 年 3 月 12 日，IBM 的股价和市值达到历史最高，股价达 215 美元，市值超过 2000 亿美元。为什么是 3 月 12 日呢？因为在这时，已经基本确定 IBM 收购 Softlayer，美国股市也有内幕消息啊。

2013 年 6 月，云计算的大势已经很明显，而谷歌和微软在云计算上的动作也比较明朗了。但 IBM 只愿意动用 20 亿美元为其云计算打基础，还不到 6 年前 2007 年 50 亿美元现金收购 Cognos 的一半金额。除了 Softlayer 之外，IBM 基本没有像样的针对云计算的收购，而在数据分析上的收购则有数十宗。IBM 在 2013 年对云计算的态度可以用“敷衍”来形容。

而且市场上并不缺乏收购标的，特别是 RackSpaceCloud。RackSpaceCloud 当时已经在同 AWS 的竞争中处于下风，所以在 2013 年 1 月，传言 IBM 和 EMC 联手竞购 Rackspace 后，Rackspace 股价在 2013 年 1 月 24 日超过 76 美元，市值达 106 亿美元。如今 Rackspace 的市值只剩下 28 亿美元。

EMC 显然看不上 Softlayer，没有掺乎这档子事。EMC 下属的 VMware

随后推出了自家公有云 vCloud Air。

IBM 在 2016 年之前的 10 年在商业分析上花费至少一百亿美元，而在威胁其数据中心软硬件和服务之根本的云计算上，只花了 20 亿美元。

而且这 20 亿美元，还是花在了一个根本不是云计算、就是一家 IDC 挂羊头卖狗肉的公司上——SoftLayer。

在 IBM 收购之前，SoftLayer 更像一家 IDC 公司而不是一家云计算公司，至少在我的印象中是这样。或许 IBM 看重的只是 Softlayer 的机房说不定。SoftLayer 跟国内顺带买卖 VPS 和云主机的 IDC 没什么区别。Softlayer 2012 年的营收约为 4 亿美元，云计算营收至多数千万美元。而 Rackspace 2012 年营收约 13 亿美元，云计算营收近 3 亿美元。Rackspace 的云计算营收只有亚马逊 AWS 的六分之一，但考虑到 Rackspace 在 IDC 和技术支持领域的实力，以及其在云计算方向能够投入的资源（2015 年 Rackspace 手上的现金约一亿美元，现金等价物约 2 亿美元），这已经是不错的结果了。彼时 2012 年，微软 Azure 的营收可能远低于 Rackspace，Rackspace 作用云计算基础设施服务老二的位置。

2012 年，Softlayer 的营收只有 Rackspace 的三分之一，而 SoftLayer 的云计算营收只有 Rackspace 的十分之一，SoftLayer 的云计算能力和潜力只有 Rackspace 的百分之一，如果 Rackspace 价值 70 亿 ~100 亿美元（当时业界收购传闻价），SoftLayer 怎么值得到 20 亿美元。SoftLayer 就应该按照 IDC 行业估值，价值 11 亿 ~12 亿美元。

2012 年，在云计算方面，SoftLayer 不要说比 Rackspace 差一个量级，就是比 Gogrid、Joyent 都要弱许多，就是数据中心多一点而已。IBM 在云计算上最重要的一笔收购完全是狗屎。IBM 或者被忽悠了或者图便宜，这笔交易能带来 IDC 业务上的营收，却带不来云计算业务能力的提升。

在最近的两年，IBM 除了 Softlayer，也着重打造了 CloudFoundry 的 IBM 版 -BlueMix，以及 OpenStack 的 IBM 版 -Blue Box。更重要的是，打造了商业分析的系列服务和交易市场。并正在推出主打容器的产品 Linux One，首次在大型机上基于 Ubuntu 发布产品，支持最高 10TB 内存。

2014 年 9 月 18 日，IBM 发布 Watson Analytics。提供自助式分析功能，包括数据访问、数据清洗、数据仓库，帮助企业用户获取和准备数据，并基于此进行分析，实现结果可视化。

2015 年，IBM 继续猛炒 Watson 和人知技术。2011 年，Watson 参加综艺节目危险边缘（Jeopardy）来测试它的能力，这是该节目有史以来第一次人与机器对决。当年 2 月 14 日至 16 日广播的 3 集节目中，Watson 在前两轮中与对手打平，而在最后一集里，Watson 打败了最高奖金得主布拉德·鲁特尔和连胜纪录保持者肯·詹宁斯。Watson 赢得了第一笔奖金 100 万美元，而肯·詹宁斯和布拉德·鲁特尔分别只有 30 万和 20 万。Watson 4TB 磁盘内，包含 200 万页结构化和非结构化的信息，包括维基百科的全文。在比赛中 Watson 没有链接到互联网。

2016 年 2 月 5 日，IBM 一口气发布了 4 款数据分析产品：IBM Compose Enterprise、IBM Graph、IBM Predictive Analytics 和 IBM Analytics Exchange。

这一切都要归结到商业分析。

商业分析，是 IBM 10 多年打造的业务，既是 IBM 的强项，被 IBM 寄予厚望。云计算？从心里根本就不想搞，但不搞还不行。但或许是想搞，但翻来覆去没搞出个名堂，或者不知道怎么搞。

IBM 依旧是大象，只是已经不是一只能够轻盈起舞的大象。而是一只身陷泥潭，正在挣扎着往岸边挪，却还要假装舞姿轻盈而潇洒。只是从哪

边上岸，可能没看清楚，能否爬上岸，也更是个未知数。

把云计算和商业分析结合起来，或许是一条路子。IBM 15 年打造的商业分析业务，显然甩亚马逊和微软几条街。IaaS 和基础型 PaaS 既然很难撬动，那就摆摆花架子，跳过这个层面的血腥竞争。用商业分析（包括人工智能、数据分析）来搞定开发者、企业，打开应用型 PaaS 的局面，最后形成全栈云计算平台。

IBM 的云计算战略，效果如何？目前不知道。IBM 的云计算营收比微软还让人晕。号称上百亿，自己说自己都不信。IBM 的公有云，2015 年最多在 7 亿美元上下。

IBM 的云计算战略，前景如何？下半部和 Oracle、Google 的一起来说。

IBM、Google、Oracle 三巨头的公有云之殇（下）

作者 刘黎明

IBM 的云计算：收购、商业分析之外，还有 PaaS、合作、创新

IBM 的收购和商业分析我们在上面说了不少。大体上说，只有对 SoftLayer 的收购算是纯粹云计算意义上的目标。只是这个标的，当时可能合算，但并非合理。我认为收购商的短版，可能是高层对云计算的重视问题，但更可能是并购团队对云计算的了解和认识问题。干了十几年了硬件、企业软件收购，突然做公有云领域的收购，确实难度大。

商业分析是 IBM 多年来积攒下来的老本，用在云计算上也算一举两得。避免了和市场上的巨头刚正面，而且现在深度学习、人工智能、大数据分析正火，可以说得上是以己之长攻彼之短。

但是如果 IBM 只有这两招，IBM 就不是能跳舞的大象了。他也不可能在企业 IT 做大哥做半个世纪之久。

IBM 在公有云上起步可能是巨头里比较晚的，在 2013 年 6 月收购 SoftLayer 后才算正式开始。

IBM 这次绝对是有雄心、有魄力、有耐心的。我所说的全力以赴，放在 IBM 身上并不为过。而创业公司们，如果没有 IBM 这样的灵活、耐心，不要说与巨头竞争，如果还能活得下去，那一定是撞了大运。我，当然还有读者们，都不喜欢看史料、证明、罗列，但特么不看看，你还真不知道

IBM 下了多大功夫。

2014 年之后，IBM 在商业合作和创新、开放上的阵仗，绝对不输任何公司，也不输于我们想象到的任何程度。IBM 不是只会利用历史积累，它还会开拓、创新。

2014 年 8 月，IBM 发布 CloudFoundry 的 IBM 版公有云 -BlueMix，号称投资 10 亿美元，这是 IBM 收购 Softlayer 起步 IaaS 后，大搞 PaaS 的开始。自带四种 Bludpacks，分别是 Libertyfor Java，Node.js，Ruby on Rails，Ruby Sinatra，具备 DevOps 能力，并实现了物联网 API、Watson 人工智能的语义识别、分析、预测的 API。在 2015 年 12 月 2 日，推出 BlueMix Local（本地版本）。中间还发布了 dedicated（专属版本）。

2014 年 9 月 18 日，IBM 发布 Watson Analytics。提供自助式分析功能，包括数据访问、数据清洗、数据仓库，帮助企业用户获取和准备数据，并基于此进行分析、实现结果可视化。

从 2014 年起，IBM 下属的 IaaS Softlayer 开始强调 bare-metal 能力。这当然是把 IBM 对于数据中心的管理能力与 Softlayer 结合起来，形成的新的云服务特点。这就叫差异化。

2015 年年中，IBM 发布 Linux One，首次在大型机上基于 Ubuntu 发布产品，支持最高 10TB 内存。Ubuntu 作为服务器操作系统，在 IT 运维工程师里是有争议的。但为了争取开发者，IBM 和最流行的 linux 发行版 Ubuntu 走到了一起。

2015 年 6 月 4 日，IBM 收购 Blue Box ，推出 OpenStack 业务。思科公司于前一天收购 OpenStack 厂商 PistonCloud。

2016 年 02 月 18 日，IBM 宣布推出区块链服务平台，瞄准金融业和运输业。这比微软推出区块链即服务晚了三个月，但这绝对是区块链的早期。

2016 年 2 月 5 日，IBM 一口气发布了 4 款数据分析产品：IBMCompose Enterprise, IBM Graph, IBM Predictive Analytics 和 IBMAnalytics Exchange。

2016 年 2 月，IBM Cloud 宣布支持 Swift，这是第一个支持 Swift 的公有云平台。苹果副总裁为其站台。IBM 是企业级老大，苹果是智能手机老大，以前大家各干各的，但是现在，走到了一起。

Swift 是苹果在 2014 年推出的一款编程语言，推出之后得到了开发商们的热烈追捧。不热捧不行啊，因为苹果手机是大部分移动应用开发商大部分营收的来源。

苹果在 2015 年 12 月又向前迈出一大步，将 Swift 转为开源语言。这也进一步提升了 Swift 的成长性和受欢迎程度，Swift 也因此迅速攀升到 GitHub 受欢迎程度排行榜的顶部。

IBM Cloud 可以为企业应用开发者提供一整套与在 Swift 中编写软件有关的工具。以前，任何开发商想要在云服务器上运行 Swift 都必须切换语言，现在不需要了。

这显然少不了苹果的大力协助。苹果和 IBM 还真是有缘。大家还记得九十年代，IBM 和苹果宣布建立合资公司，开发和销售 PowerMac，对抗 wintel 吗？

这一次，历史重演。只是对抗的对象是云计算的巨头 - 亚马逊。敌人的敌人就是朋友，何况以前还是共同战斗过。据说在这次合作之前，IBM 已经很挺苹果了，人手一个 Mac。

苹果已经打算大幅减少在亚马逊 AWS 上的基础设施，那么苹果可能使用自家数据中心，也可能转而使用 IBM 的云服务。毕竟，苹果与亚马逊的竞争领域不止一个，而与 IBM 基本没有直接竞争。

苹果可是个大腿啊，一年几十亿美元的基础设施预算。

还是 2016 年 2 月 22 日，IBM 一年一度的 InterConnect 大会上宣布，与 VMware 达成战略合作。双方已联合设计了一套架构和一系列云端产品，可以让客户在 IBM 的混合云上运行 VMware 的产品。

根据合作声明，未来两家公司将向客户推荐对方的产品。IBM 提供的资源是其在全球搭建的 45 个云数据中心，IBM 将向这些数据中心的客户提供 VMware 的产品。VMware 分享的也是用户资源。VMware 的传统客户可以选择从私有云上迁移至 IBM 的混合云，而不需要担心 VMware 软件的兼容问题。

IBM 虽然一直和 VMware 有虚拟化产品的合作，但在竞争关系更大。IBM 虽然一直没有主打的虚拟化产品，但一直没有停止开发自有虚拟化产品的行动。而且 IBM 属于 Power 服务器阵营，而 VMware 和 Intel 属于 X86 阵营。

以前双方就是亦敌亦友，以后也还是。但是，在公有云的压力下，双方终于携手合作。双方将自己的产品、资源、客户放开了，客户自己选择。

家底、青梅竹马的朋友的大腿、亦敌亦友的伙伴的绝技、成熟的产品、新的晃眼的技术，IBM 全亮出来了。这都是为了公有云，而且 IBM 的嗓门大可不是盖的。IBM 云上的大事小情，IBM 的公关部门绝对能搞得天下皆知。这方面，IBM 绝对不孤单，Oracle 就是一个很好的狐朋狗友。

Oracle 的云计算，一直说在憋大招，但我看到的不是捆绑销售就是收购

Oracle 一直在憋大招，欲动摇云计算格局。与 Oracle 对待云计算的态度不同的是，Oracle 的云计算市场地位还没有逆袭。它的大招肯定不是收购，那是捆绑销售吗？当然不是。云计算浪潮里，Oracle 沉睡的狮

子还是迟到者？

在 2008 年 9 月，Oracle CEO 埃里森说，云计算所代表的东西就是现在所做的一切，我们现在所做的一切都是云计算。并说，云计算就是胡言乱语，是废话，是愚蠢。他也不认为云计算能带来什么新的不一样的东西。

在 2012 年的一场财务分析对话中，埃里森说是他发明了云计算，然后被媒体一顿批。但是他是网络 PC 的早期支持者，一种瘦客户端。埃里森有时会有灵光乍现的时候，但也有固执的时候。比如，对待初期的云计算，他是反对的。但是企业家就是企业家，很快就回过神来。虽然回神的过程有点漫长，但回神之后，终于就将固执转换成魄力。只是这个过程经历了 5 年之久，云计算形势已经很不利。

在 2010 年 Salesforce 以 2.12 亿美元收购了 heroku 之后，Oracle 收购了其竞争对手 EngineYard。但这起收购没有全面看起 Oracle 的云计算道路。

在这之后的四五年里，Oracle 看着自己的 Java 成为其他云的重要运行环境，帮助其他云运行 weblogic 和 oracle 数据库，可以说为其他云的发展立下了汗马功劳。2015 年，Oracle 终于决定要亲自上阵，搞自家的云计算，从 IaaS 到 PaaS 到 SaaS。

2015 年 12 月，Oracle 发布的第二季度财报显示收入下滑 6% 为 89 亿美元，而净收入下滑 12% 为 25 亿美元。但是，公司的高层却希望通过强调云计算收入增长 26% 来摆脱这些数字的影响，云计算收入达到了出色的……6.49 亿美元。

Oracle 首席技术官兼主席拉里·埃里森表示：“我们的目标仍然是在这个财年里销售和预订超过 15 亿美元的、新的 SaaS 和 PaaS 业务。”埃里森表示：“这可是相当多的 SaaS 和 PaaS 业务，比任何其他的云服务

提供商的都要多，甚至比 Salesforce.com 还要多。”

没错，是不是真的比 Salesforce.com 可不知道。但 Oracle 在 2015 年，有认真的传闻，准备以 440 亿美元收购 Salesforce。

同时，市场上传来了大量的 Oracle 声音。有用后发制人理论为 Oracle 打气的，有用企业级客户保有量和忠诚度证明 Oracle 云计算潜力的，以及 Oracle 在亚太或全球招聘 1000 名销售的，有说 Oracle 正在打造秘密武器的。

2015 年秋季，埃里森说 Oracle 是一家云计算公司，再也不要拿 Oracle 和 IBM、SAP 对比了。意思是以前 Oracle 最大的竞争对手是 IBM 和 SAP，但这两家公司现在不是云计算公司了。但随后媒体巴拉了一下，发现 Oracle 看不上眼的 SAP 在年股价和营收上的表现要好于 Oracle。自 2010 年以来，SAP 股价上涨了 77%，而 Oracle 上涨了 60%。2015 自然年 Oracle 云计算营收 23.63 亿美元，比上年 18.59 亿美元增加 27%，SAP2015 年云计算营收 2.286 亿美元，比上面 1.087 亿美元增加 110%。

当然，这只是媒体为了反击一下埃里森，SAP 财报统计的云计算并不一定是常规意义上的公有云。就算是真的，如果一家公司财报稳健，但是失去了未来的可能和希望，又有什么值得称道呢？牛逼的公司，通常是管理层，大部分时候是 CEO 或创始人，能给公司指引未来的方向并向着这儿目标前进。

2015 年可以说是 Oracle 在云计算上奋发图强的一年。

2015 年 7 月，埃里森代表 Oracle 一口气发布了 Oracle Marketing Cloud、OracleIntegration Cloud、Oracle Analytics Cloud、Oracle Commerce Cloud、Oracle Mobile Cloud Service 五款产品，并在 10 月发布了 Oracle Manufacturing Cloud。营销、分析、移动、制造，自己

有的没的，擅长的不擅长的，都上云了。

在 2015 年 Oracle OpenWorld 大会上，10 个主题演讲里，8 个和云计算有关，四个是关于 Oracle Cloud。并且，终于在 AWS 发布 EC2 9 年之后发布了 Oracle Elastic Compute Cloud。

直到 2015 年 12 月，Oracle 收购了一家名为 StackEngine 的创业企业，这家位于奥斯汀的公司只有五名员工，干的是容器的事。我估计基本上是个人才收购，Google、微软、亚马逊的云都有容器服务了，Oracle 的也得有啊。这次收购所体现的魄力、眼光，是 2015 年 Oracle 云战略的缩影。

2015 年 2 月 25 日，Oracle 出价 5 亿美元，收购了初创企业 RavelloSystems。Ravello Systems 创建于 2011 年，总部位于美国加州帕洛阿图（Palo Alto），这里距离 Oracle 总部所在地红杉城（Redwood City）并不远。Ravello 帮助客户快速简便地将复杂应用程序移动到云端。这家创业公司的客户包括 Arista、博科通讯（Brocade）、红帽（Red Hat）、SUSE 和塞门铁克（Symantec）。

2015 年 2 月中旬，Oracle 搞了一个云计算战略发布会，即宣布了雄心勃勃的产品组合，Oracle 选择了“简单”作为重点。Oracle 的简单，是指企业可以简单地部署到公有云私有云混合云上。简单并不是现在云计算的重点和热点，但却是 Oracle 想要与别人竞争的点。这可是智慧啊。好好体会。

忘了埃里森曾说的云计算是愚蠢的玩意这回事吧。2015 年，Oracle 说 Oracle 搞的是企业级云计算，AWS 搞的是小公司的玩具云计算。

企业级云计算？很熟悉吧。哪里有什么企业级云计算，那不过是传统 IT 企业用来唬人的而已。所有声称提供企业级云计算业务的公司，都已经鸣金收兵或正准备跑路了。Oracle 还说自己是企业级云计算，还能搞

点别的词儿吗？

当然有别的词儿！ Oracle 的营销机器开动起来，也不输于 IBM。有人专门为 Oracle 的云计算搞出了一个 Alpha 和 Omega 理论。

Alpha 理论是说，市场上有一个先行者，已经获得了相当的市场份额，快速创新，且维持较低的价格。

Omega 理论是说，市场仍然留有空间给一个强大的竞争中位置，这个竞争者有强大的财力、强大的留住现有企业客户的能力和一个合适的差异化策略。

Alpha 是指 AWS，Omega 是指 Oracle。如果按 Omega 理论，Oracle 确实比 Google，甚至比 IBM、微软更有优势。

Oracle 当然有足够的财力，其数据库统治地位当然能够锁定和留住现在的企业客户。最大的疑问则在于它是否能够构建一个有竞争力和差异化的策略挑战 AWS。

Oracle 的大招可能是基于容器的混合云。基于容器的混合云将允许客户将私有负载轻松转移到公有云上，甚至将其他云上的负载转移到 Oracle 的云上。特备的，能将 Oracle 数据库迁移到 Oracle 的云上。

不要再认为 Oracle 只是一家数据库公司，Oracle 的产品线涵盖硬件、软件、企业应用、中间件、操作系统等，几乎包括企业 IT 的整个技术堆栈。Oracle 的核心产品显然仍然拥有强大的统治力和粘性，可以保证足够的资源投入到云计算战争中去。

现在的 Oracle，誓要成为一家云计算公司。Oracle 对云计算的态度和投入，绝对是首屈一指的。以至于其他的 IT 公司，都相比而逊色。

Oracle 起晚了，但还是赶上了早集。Google 起了个大早，却和 Oracle 同一趟车去赶集。

Google 云携巨款亮绝技：价格和技术

没人怀疑谷歌的资金实力、基础设施规模、创新能力、技术实力。谷歌依然最赚钱的互联网公司，是除苹果意外最赚钱的 IT 公司，它运营者世界上最大规模的 IT 基础设施。三大核心技术 MapReduce、GFS 和 BigTable 形成的论文八年前就已风靡网络，至今仍被被奉为圭臬。

Google 非常有进取心：不断进入新领域，从气球 Wifi、谷歌眼镜，到无人驾驶，无不是被认为创新者中的极客；要做一件事情也很投入，比如视频，YouTube 很多年都没有盈利但仍能获得支持，再比如社交领域，从 google wave 到 google+，及时不能丝毫阻挡 facebook 也要三番五次尝试；有技术，从早期的三大技术论文，到近些年的 android、html5，都是行业奠基性的工作。

如果 Google 要在某一并未形成未定格局的市场挑战一家并不是那么财大气粗的公司，比如亚马逊，会有什么问题吗？没问题，只是 Google 犯了一些错误：早期产品路线有问题，后期魄力不够。这可能是战略视野问题，也可能是能力问题。

谷歌的嗅觉和行动也并不算太迟钝。在 2008 年，也就是微软发布 Azure 的两年前，谷歌就发布了 Google App Engine。可以说 Azure 就是沿着 GAE 的思路来的，Azure 就是 windows 版的 GAE，GAE 也可以认为是 Linux 版的 Azure。相似的思路，迎来了相似的开局：开发者并不认同 PaaS。

谷歌的云计算主管曾经说：如果 GAE 是一家创业公司，它一定是硅谷的耀眼明星。很可惜的是，在那段时间里，GAE 在 AWS 的快速增长面前显得苍白无力。

更为可惜的是，Azure 在 2011 年就加入 VM 角色切入 IaaS 层面的服务，

而谷歌在 2012 年 6 月才推出 ComputeEngine 切入 IaaS。这中间花了四年，比晚两年出生的 Azure 还慢了 1 年。我不知道纳德拉对技术的涉猎有多深，但谷歌的技术人员显然对 GAE 情有独钟，因为 GAE 完全就是将三大论文里的三大核心技术商业化了，多么理想的技术平台！然而，这不是大部分客户需要的。

谷歌显然有相当一批开发者拥趸，也包括一批技术极客。谷歌发表的技术论文和开源项目，令这些人相比微软，更为相信谷歌。尽管纳德拉时代的微软对开发者比以前有好多了，但在这些人眼里，微软就是专制、封闭的代表，而谷歌则是自由、开源的代表。但是，喜欢谷歌的技术和理念，与使用谷歌的云计算基础设施服务，完全是两码事，可以分得很开。

2015 年，Leo Sun 发表了一篇标题为“谷歌和亚马逊：谁能赢得云冷战”的文章。文章认为谷歌和亚马逊正在进行异常关于旧数据的冷战。看着标题是不偏不倚，但内容表明作者更看好谷歌。谷歌的 Nearline 可以再三秒内读出旧数据，而亚马逊 AWS 的 Glacier 需要三到五个小时。谷歌能做到这样是因为谷歌大新数据和旧数据放在一个系统里，而亚马逊 AWS 在放在两个系统里。

作者认为这让 Google 有了切入企业级冷数据和混合云市场的优势，并且 Google 能将 IaaS 和 PaaS 服务于 Google 的 android 和 apps 生态系统对接，让终端用户和企业在使用应用的时候，使用 IaaS 和 PaaS 服务。作者认为这只是 Google 具有的高性价比技术带来优势的一个例子，辅以 Google 更广泛的生态链支持，这将最终能够扼杀羽翼未丰的 AWS 的增长。

这几年来，谷歌也发布了更多的 IaaS 和 PaaS 产品，包括存储、数据库、数据分析等，并在 2013 年形成 Google Cloud Platform。

最近有分析指出，Google 定能打败亚马逊 AWS。因为一方面看，云计

算第一第一阶段是泡沫，第二阶段是是关于用云计算构建原来就在用的数据中心模型，第三阶段是利用云计算做企业原来不能构建的，新的基础设施和应用模型。比如用 10 倍于企业日常使用的基础设施，进行数据分析。亚马逊都具有第二阶段，在云计算中构建用户的数据中心的能力。但第三阶段，是关于机器学习、神经网络、人工智能的，Google 在这些方面远远将 AWS 甩在后面。其实，IBM 是不是也是这么想的呢？

Google 和微软都在搞机器翻译，这不仅需要大量的存储和计算资源，还需要算法那。Google 发布了图片识别和图片地理位置识别的 API，这需要对 1.26 亿张带有地理位置信息的图片进行学习和训练。

谷歌确实吸引了诸如可口可乐、百思买这样的客户。最近的大事件，则是成为 Google 的最新大客户。这是 Google 对亚马逊 AWS 和微软的胜利，但也反映了谷歌的企业级客户的缺乏。

IBM、Oracle 及 Google 谁的公有云先轰然倒下

IBM、Oracle、Google 这几个家伙真正紧张起来，是在 2015 年 4 月，AWS 发布财报后。

这之后，IBM 的投资者们和高管团队都认为云才是扭转公司不利局面的关键。这是不是晚了点？永远都不晚。

根据 Synergy Research Group 的市场调研结果，亚马逊 AWS 和微软 Azure2014 年合计占据 IaaS 和 PaaS38% 的市场份额，2015 年合计占有 40% 的市场份额。

IBM 是最晚进入主要公有云服务商名单的，也是这个名单里年度营收增速最慢的，基本等于市场整体增速 -40%。

收购完 SoftLayer 这么多年，快三年了，IBM 还没有理清楚 SoftLayer 与 IBM 云计算业务和平台之间的关系。Softlayer 自行其是，

朝着 IDC+IaaS 平台的路前行，而 IBM 在朝着数据分析 PaaS 的路狂奔。IBM 和 Softlayer 的云名义上都属于 IBM，但技术和运营上各自独立，没有融合成一体，分开来看又缺胳膊少腿。

这么多年，IBM 没有搞出一个像样的虚拟化、私有云、混合云软件，我也是醉了。还要求助 OpenStack，掺乎 OpenStack 也就算了，居然没搞出个所以然来，还需要收购搞 Openstack 的 Blue box，或许又是去买客户的。

收购管用吗？ 200 多亿美元的收购，IBM 的年度营收增长几乎为零。关键是，这些收购来的业务不要说盈利，收购后历年总计营收达到 200 亿美元都有问题。

IBM 的收购战略显然存在问题。一是收购是否能够良好整合、是否能够打造核心产品和产品群，进而打造公司核心竞争力。二是收购标的是否合理，这是执行层面的问题，但也同样关系到收购战略的成败。

IBM 的商业分析为核心的战略，可能也存在问题。

在云计算演义（序）中，我们提到这样一段话：

首先，我们要意识到，传统企业 IT 的高毛利率将不复存在；

其次，云计算的高毛利率也不会存在。

这是时代赋予云计算的使命。IT 基础设施和大部分应用都将成为普及性社会基础服务，将成为日用品，将从企业的核心竞争力变成标准配置。

这可能是部分公司和从业人员的不幸，但这是整个信息产业和全社会发展的利好。

那么，IBM 的总体战略可能也存在问题，即业务转向高价值业务。如果高价值等同为高利润，就如同过去 15 年 IBM 所做的那样，抛弃低利润的产品，可能也存在问题。

有人说，颠覆式创新有三种：第一种是彻底的发明。但现在的信息行业，更多的是另外两种：用户体验的创新；商业模式的颠覆。也有人说，颠覆式创新最后颠覆的全是人性，人性有两个最基本的东西：简单和便宜。

IBM 百年来一直在做的正是第一种颠覆式创新——彻底的发明。但不幸的是，因为行业发展更加成熟，彻底的发明越来越难也越来越少。近二十年来，IBM 也少有彻底的发明。

用户体验的创新商业模式的颠覆，包括简单和便宜，并不是 IBM 和 Oracle 留给人们的印象，也不是它们的强项。而云计算，需要简单和便宜。

这决定了 IBM 和 Oracle 做公有云，是个巨大的矛盾。云计算业务扩张所要求的大规模、简单、便宜、快速反应，并不是 IBM 和 Oracle 的强项，更直白一点，也不是 IBM 和 Oracle 的基因所在。除非 IBM 和 Oracle 能变革自己的文化和基因，涅槃之后，必成凤凰。

2014 年年中，IBM 曾被迫加入微软、亚马逊、谷歌的云计算价格战。在云计算格局未稳定的时候，价格战根本不会停止。这将大大损耗 IBM 本已脆弱的财务报表。而 IBM 是非常在意和倚重财务报表的，IBM 和 Oracle 也是几个大公司里财务报表压力最大的。

亚马逊虽然也有财务压力，但谁让它的报表一直就那个样呢？华尔街也习惯了。另一个，贝佐斯总有办法说服投资人相信他的长期投资策略。最后，也是更重要的是，亚马逊的营收保持着相当幅度的增长，这可以很大程度上抵消净利润上的不足。

Oracle 全力转云并不是心血来潮，而是必须赢得这一战。当所有企业都将工作负载迁移到云上，数据库将变成一个普通商品，有很多选择可以满足用户需求，到那时客户将不对 Oracle 存有依赖，Oracle 也将失去其留住和转化客户的能力，包括随之而来的财力。

事实上，AWS 已经在这么做了。AWS 的数据库迁移工具可以帮助数据库管理员脱离 oracle 数据库。这当然是在动摇 Oracle 的核心权力。

我们再来看看专为 Oracle 定制的 Alpha 和 Omega 理论。Alpha 理论中的领先者有着明显的优势：

规模效益。先行者拥有更大的规模，因而可以满足更大用户的需求，同时能够降低采购和运营成本，从而提供更低的价格。

生态系统。先行者的先发地位，和更大规模的用户群体，肯定能够吸引软件开发商、系统集成商、服务提供商在其平台上展开业务，从而围绕这个平台形成生态。这种生态不是某个技术和项目的开发者社区可以比拟的，比如 OpenStack 生态，就难以在商业上和 AWS 生态相比拟。

开发者雪球效应。先行者形成产品群以后，其使用也需要较高的技能，其前期用户群具备了这些技能后，将通过同事关系和社区关系，带动更多开发者更容易掌握这个平台所需的技能。

进入壁垒。这不仅包括研发和上线所需的一次性人力和数据中心投入，还包括维持创新、维护产品和数据中心所需的持续性投入。特别是在激烈竞争期，由价格战的情况下，这将使巨大的财务黑洞。

但是 Omega 不是么有机会，我们可以笼统称之为后发优势。后发本身是一个巨大的劣势，要想拥有优势，除非：

新一代的技术。技术代际之间的差别越大，率先采用新技术的后发者越有机会挑战先行者。比如数字通信技术较之于模拟信号技术，比如 4G 至于 3G。即使先发者意识到新技术的必要性，也会因为历史业务的延续性和更新换代的巨大成本，而丧失合适的时间窗口。

明显的差异化。快速的追随者，能够从先行者身上吸取经验教训，以便指定一个明显的差异化策略吸引用户。

很显然，维持领先地位比屌丝逆袭要容易得多得多。不管 Oracle 的 Omega 策略是什么，是不是容器基础设施都好，Oracle 的机会随着时间的流逝，将越来越小。

IBM 和 Oracle 都面临财务压力，特别是他们是在左右手互博。就好比把左兜里的钱装到右兜里来，但换个兜后，钱却少了。他们要把现有客户转化成云客户，就是这个矛盾。而且不转还不行，不转，就全跑别人兜里去了。少就少点，但总比没有强啊。

Google 最逍遥，因为这些云计算业务都是从别人兜里掏来的，根本无损于 Google 现有的业务和营收。而且在现在的业务依然很赚钱的情况下，它干云计算并没有那么强大的动力，更没有什么压力。这就是 Google 总感觉有点不紧不慢的原因。这也是他最危险的地方。

没有人愿意把自己的基础设施，放在一个对云计算不够有兴趣，或者对云计算三心二意的厂商手里。但 Google 还是很努力的，从上面的文章，以及请来花一亿多美元请来 VMware 创始人兼前 CEO 来掌舵云计算来看，Google 还是个努力的同学。

IBM、Oracle、Google 都很卖力，各有盘算，但都能成为头牌吗？算上亚马逊 AWS 和微软 Azure，显然不能都成为头牌。

谁最先倒下？最先关闭云计算服务？可能是谷歌，谷歌看起来战略地位最有利，实力最强大，但最先倒下的可能就是它。谷歌想干的大事太多了，云计算只是其中一件。

然后是 Oracle，Oracle 的数据库在新型数据库面前，已经陷入绝对的衰退。用户有很多商业的、开源的、私有的、公有云的数据库可以选择，而且这些数据库很多都已经更具备或正在具备企业级水准。云计算对数据库的影响在最近几年才明显体现，因此 Oracle 发力的时间太晚了，其云

计算发展速度很可能赶不上其他重要业务的衰退速度，此时它对客户的制力和可投入的资源将迅速下滑。

而 IBM，很可能是三家公司中生命力最强的。云计算对 IBM 的打击最直接，IBM 的起步也晚了些，但 IBM 在企业 IT 的根基比谷歌和 Oracle 都要深，大象的绝地反击应该是最有力的。IBM 运筹得当，甚至能挑战微软现在的老二位置。

我们当然不希望他们任何一个服务商彻底倒下，那样损失的将是用户。他们或许会在某一天关闭云计算服务，或许会成为二流云计算服务商。如果果真如此，那是忽视创新、因循守旧、缺乏睿智应付的代价。

北京 Beijing | 伦敦 London | 纽约 New York | 里约热内卢 Rio De Janeiro | 旧金山 San Francisco | 圣保罗 Sao Paulo | 上海 Shanghai | 东京 Tokyo

QCon [上海站]

全球软件开发大会2016 | 宝华万豪酒店 10.20~22

主办方 Geekbang. 极客邦科技 InfoQ

《浅谈代码复用攻击与防御》

陈 平

点融网信息安全负责人

《大型企业云平台架构和关键技术实践》

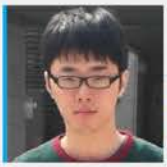
苗彩霞

华为软件云平台资深架构师

《从近年来中美技术创业对比看创业新机会》

赵 斌

声网Agora.io创始人&CEO



《Weex极致性能优化》

冯成晓（隐风）

阿里巴巴无线技术专家

《Java模块化技术演进和对现有应用微服务化的意义》

张建锋

永源中间件共同创始人

《apk沙箱技术在平台型app中的架构实战》

黄明登

科大讯飞架构师

《深度学习框架的性能优化及其在医药行业的应用实践》

叶 军

英特尔资深架构师

《打造万亿级别的数据流水线》

Steven Wu（吴震）

Netflix软件工程师

《如何打造大规模互联网企业的监控告警平台——以携程hickwall为例》

唐锐华

携程高级工程师

《JVM虚拟化—— 重新定义Java容器热部署资源管理机制》

陆传胜

阿里巴巴技术专家

《深度学习技术在图片搜索与图像搜索上的实践》

周泽南

搜狗资深研究员

《转型路上，苏宁技术的砥砺与涅槃》

乔新亮

苏宁云商IT总部执行总裁助理

《冰与火之歌：企业安全的攻与防》

凌 云

携程信息安全总监

《构建规模化的企业级风险感知体系》

张天琪

上海斗象科技联合创始人兼CTO

《同程旅游微服务架构设计实践》

王晓波

同程旅游首席架构师

8折优惠 8月21日前报名 立减1360元

扫码关注QCon官微，了解大会最新动态

InfoQ 中文站 2016迷你书

架构师 月刊 2016年7月

本期主要内容：现在Google制造自己的芯片，Intel要发疯，微软开发团队的DevOps实践启示，Rust 1.0发布一周年，发展回顾与总结，微服务与服务团队在Amazon的发展，微服务架构：Kafka的崛起微服务架构终极探讨，从“人事钱心”四个方面，讲述跨过技术创业的那些坎儿。

云生态专刊 08

《云生态专刊》是InfoQ为大家推出的一个新产品，目标是“打造中国最优质的云生态媒体”。

顶尖技术团队访谈录 第六季

本次的《中国顶尖技术团队访谈录》·第六季挑选的六个团队虽然都来自互联网企业，却是风格各异。希望通过这样的记录，能够让一家家品牌背后的技术人员形象更加鲜活，让更多人感受到他们的可爱与坚持。

架构师特刊 **揭秘京东618背后的技术力量**

每年6月18日是京东店庆日。在店庆月京东都会推出一系列的大型促销活动，以“火红六月”为宣传点，其中6月18日是京东促销力度最大的一天。一度将京东618促成与双11遥相呼应的又一大全民网购狂欢节。